UNIVERSIDAD AUTÓNOMA DE MADRID

Diffusion Methods and Applications

by

Ángela Fernández Pascual

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the

Escuela Politécnica Superior
Departamento de Ingeniería Informática

Under the supervision of
José Ramón Dorronsoro Ibero
and
Julia Díaz García

June 2014

Prediction is very difficult, especially about the future.

Niels Bohr

Abstract

Diffusion Methods and Applications,

by Ángela Fernández.

Big Data, an important problem nowadays, can be understood in terms of a very large number of patterns, a very large pattern dimension or, often, both. In this thesis, we will concentrate on the high dimensionality issue, applying manifold learning techniques for visualizing and analyzing such patterns.

The core technique will be Diffusion Maps (DM) and its Anisotropic Diffusion (AD) version, introduced by Ronald R. Coifman and his school at Yale University, and of which we will give a complete, systematic, compact and self-contained treatment. This will be done after a brief survey of previous manifold learning methods.

The algorithmic contributions of the thesis will be centered in two computational challenges of diffusion methods: the potential high cost of the similarity matrix eigenanalysis that is needed to define the diffusion embedding coordinates, and the difficulty of computing this embedding over new patterns not available for the initial eigenanalysis. With respect to the first issue, we will show how the AD set up can be used to skip it when looking for local models. In this case, local patterns will be selected through a κ-Nearest Neighbors search using a properly defined local Mahalanobis distance, that enables neighbors to be found over the latent variable space underlying the AD model while we can work directly with the observable patterns and, thus, avoiding the potentially costly similarity matrix eigenanalysis.

The second proposed algorithm, that we will call Auto-adaptative Laplacian Pyramids (ALP), focuses in the out-of-sample embedding extension and consists in a modification of the classical Laplacian Pyramids (LP) method. In this new algorithm the LP iterations will be combined with an estimate of the Leave One Out CV error, something that makes possible to directly define during training a criterion to estimate the optimal stopping point of this iterative algorithm.

This thesis will also present several application contributions to important problems in renewable energy and medical imaging. More precisely, we will show how DM is a good method for dimensionality reduction of meteorological weather predictions, providing tools to visualize and describe these data, as well as to cluster them in order to define local models.

In turn, we will apply our AD-based localized search method first to find the location in the human body of CT scan images and then to predict wind energy ramps on both individual farms and over the whole of Spain. We will see that, in both cases, our results improve on the current state of the art methods.

Finally, we will compare our ALP proposal with the well-known Nyström method as well as with LP on two large dimensional problems, the time compression of meteorological data and the analysis of meteorological variables relevant in daily radiation forecasts. In both cases we will show that ALP compares favorably with the other approaches for out-of-sample extension problems.

Resumen

Métodos de Difusión y Aplicaciones, *por Ángela Fernández.*

Big Data es un problema importante hoy en día, que puede ser entendido en términos de un amplio número de patrones, una alta dimensión o, como sucede normalmente, de ambos. Esta tesis se va a centrar en problemas de alta dimensión, aplicando técnicas de aprendizaje de subvariedades para visualizar y analizar dichos patrones.

La técnica central será *Diffusion Maps (DM)* y su versión anisotrópica, *Anisotropic Diffusion (AD)*, introducida por Ronald R. Coifman y su escuela en la Universidad de Yale, la cual va a ser tratada de manera completa, sistemática, compacta y auto-contenida. Esto se llevará a cabo tras un breve repaso de métodos previos de aprendizaje de subvariedades.

Las contribuciones algorítmicas de esta tesis estarán centradas en dos de los grandes retos en métodos de difusión: el potencial alto coste que tiene el análisis de autovalores de la matriz de similitud, necesaria para definir las coordenadas embebidas; y la dificultad para calcular este mismo *embedding* sobre nuevos datos que no eran accesibles cuando se realizó el análisis de autovalores inicial. Respecto al primer tema, se mostrará cómo la aproximación AD se puede utilizar para evitar el cálculo del *embedding* cuando estamos interesados en definir modelos locales. En este caso, se seleccionarán patrones cercanos por medio de una búsqueda de vecinos próximos (κ-*NN*), usando como distancia una medida de Mahalanobis local que permita encontrar vecinos sobre las variables latentes existentes bajo el modelo de AD. Todo esto se llevará a cabo trabajando directamente sobre los patrones observables y, por tanto, evitando el costoso cálculo que supone el cálculo de autovalores de la matriz de similitud.

El segundo algoritmo propuesto, que llamaremos *Auto-adaptative Laplacian Pyramids (ALP)*, se centra en la extensión del *embedding* para datos fuera de la muestra, y se trata de una modificación del método denominado *Laplacian Pyramids (LP)*. En este nuevo algoritmo, las iteraciones de LP se combinarán con una estimación del error de *Leave One Out CV*, permitiendo definir directamente durante el periodo de entrenamiento, un criterio para estimar el criterio de parada óptimo para este método iterativo.

En esta tesis se presentarán también una serie de contribuciones de aplicación de estas técnicas a importantes problemas en energías renovables e imágenes médicas. Más concretamente, se muestra como DM es un buen método para reducir la dimensión de predicciones del tiempo meteorológico, sirviendo por tanto de herramienta de visualización y descripción, así como de clasificación de los datos con vistas a definir modelos locales sobre cada grupo descrito.

Posteriormente, se aplicará nuestro método de búsqueda localizada basado en AD tanto a la búsqueda de la correspondiente posición de tomografías en el cuerpo humano, como para la detección de rampas de energía eólica en parques individuales o de manera global en España. En ambos casos se verá como los resultados obtenidos mejoran los métodos del estado del arte actual.

Finalmente se comparará el algoritmo de ALP propuesto frente al conocido método de Nyström y al método de LP, en dos problemas de alta dimensión: el problema de compresión temporal de datos meteorológicos y el análisis de variables meteorológicas relevantes para la predicción de la radiación diaria. En ambos casos se mostrará cómo ALP es comparativamente mejor que otras aproximaciones existentes para resolver el problema de extensión del *embedding* a puntos fuera de la muestra.

Acknowledgements

This thesis reflects the effort of many people, and without their help, this work would have never been ended.

First of all, I should thank my supervisors, Professor José R. Dorronsoro and Dr. Julia Díaz, for their continuous support and guidance. More concretely, I owe my deepest gratitude to Julia for always offering me her help, even in the hardest moments; and to José for the lots of productive discussions that have let me grow as a researcher, for the encouragement he has always given to me and, in general, for all his inestimable help.

I have to express all my gratitude to Professor Ronald R. Coifman, who gave me the opportunity of working at his department during two research visits, and with his valuable ideas and advice, a big part of this thesis was born at Yale in those months. The opportunity of working at Yale University was crucial for this research, but it could have never been done without the continuous professional and emotional support that I always receive from my friend Neta Rabin.

I do not want to forget all my other coauthors and their contributions to our work, which are at the end reflected in this thesis.

I am very grateful to all the members of the committee and official readers of the thesis. I would like to mention all the staff from the Instituto de Ingeniería del Conocimiento and the Department of Computer Sciences at Universidad Autónoma de Madrid, especially to the Machine Learning Group.

This research would have been impossible without the support of the FPI scholarship of the Universidad Autónoma de Madrid, the Spain's scholarship TIN2010-21575-C02-01 and the UAM–IIC "Chair for the Automatic Machine Learning in Modelling and Prediction".

On a personal level, I give thanks to all my friends that have asked me about my thesis and that have followed my little steps during these years. I specially want to thank José and José Luis for their support every single day at lunchtime, to Álvaro and Jorge for being our PhD big brothers and to Lexuri, for being my every-day partner of laughs and cries.

I would also like to thank with all my heart my family, the one in which I came into the world and the one I won when I got married, for their smiles and to always raise me a laugh, for their technical support with housework and meals, for their prayers, their concerns and their affection.

And I have left for the end my gratitude to my husband, Carlos, to whom this thesis is dedicated. I should thank him all his patience, his encouragement, and his love, but especially I have to thank him his support day to day, his continuous help, his valuable advice and his multiple ideas. Half of this thesis is yours. Thank you.

Contents

Lists

Algorithms

List of Figures

List of Tables

Theorems and Similar

NOTATION

Along this thesis, in general, matrices are denoted in upper-case with bold font ($\mathbf{A}$), and its components are denoted by two subscripts (a_{nm}). Vectors appear in lower-case bold font ($\mathbf{x}$) and its components are denoted using a subscript (x_n). When a vector is decomposed into several subvectors, the bold face is maintained ($\mathbf{x}_n$). To distinguish patterns from general vectors, a superscript with parenthesis is employed ($\mathbf{x}^{(p)}$).

Constant scalars are denoted using small upper-case letters (κ), while spaces and sets use a calligraphic font ($\mathscr{S}$). Functions are denoted using a slightly different calligraphic font ($\mathcal{f}$).

All the non-standard operators are defined on their first use. Regarding the standard ones, the transpose of a matrix $\mathbf{A}$ is $\mathbf{A}^\top$, and its inverse $\mathbf{A}^{-1}$. The derivative of a scalar function $\mathcal{f}$ is just $\mathcal{f}'$, and its gradient is denoted $\nabla\mathcal{f}$ as usual. $\mathrm{E}(\mathbf{x})$ represents the expectation of variable $\mathbf{x}$.

Moreover, for readability, a relation of the abbreviations and main symbols used across this thesis is included next.

Abbreviations

AD ..... Anisotropic Diffusion.
ALP ..... Auto-adaptative Laplacian Pyramids.
AMS..... American Meteorological Society.
ARMA... AutoRegressive Moving Average.
AUC ..... Area Under the Curve.

BIC..... Bayesian Information Criterion.

CT..... Computed Tomography.
CV ..... Cross Validation.

DM . . . . . Diffusion Maps.

ECMWF . European Center for Medium-Range Weather Forecast.

GEFS . . . . NOAA/ESRL Global Ensemble Forecast System.
GFS . . . . . NOAA Global Forecast System.

HMM . . . . Hidden Markov Model.

ICA . . . . . Independent Component Analysis.
ICL . . . . . Integrated Classification Likelihood.

KDD . . . . . Knowledge Discovery and Data Mining.
κ-NN . . . . κ-Nearest Neighbors.

LE . . . . . Laplacian Eigenmaps.
LLE . . . . . Locally Linear Embedding.
LOOCV . . Leave One Out CV.
LP . . . . . Laplacian Pyramids.

MAE . . . . Mean Absolute Error.
MC . . . . . Markov Chain.
MDS . . . . . Multidimensional Scaling.
MedAD . . Median Absolute Deviations.
MedAE . . Median Absolute Error.
ML . . . . . Machine Learning.
MSE . . . . . Mean Squared Error.

NN . . . . . Nearest Neighbors.
NWP . . . . Numerical Weather Predictions.

OLS . . . . . Ordinary Linear Least Squares.

PCA . . . . . Principal Component Analysis.
PR . . . . . Precision–Recall.

REE . . . . . Red Eléctrica de España.
RLS . . . . . Regularized Least Squares.
RMAE . . . Relative Mean Absolute Error.
ROC . . . . . Receiver Operator Characteristics.
RR . . . . . Ridge Regression.

SC . . . . . Spectral Clustering.
SOM . . . . . Self-Organizing Maps.
SVM . . . . . Support Vector Machines.

TSO . . . . . Transmission System Operator.

Symbols

$\mathbf{C}$ . . . . . Covariance matrix.
χ . . . . . Flux in a diffusion process.
$\mathscr{C}$ . . . . . Set of points that define the cloud to estimate the local covariance.
c_L . . . . . Size of the cloud of points defined to estimate the local covariance.

c . . . . . Commute distance.
$\mathcal{D}$ . . . . . Diffusion distance.
$\mathbf{d}$ . . . . . Degree of the graph.
$\mathbf{D}$ . . . . . Degree matrix of the graph.

δ . . . . . Dimensionality reduction precision.
$\mathscr{D}_{\text{te}}$ . . . . . Test data set.
$\mathscr{D}_{\text{tr}}$ . . . . . Training data set.

$\mathcal{E}_{\text{mae}}$ . . . . . Mean absolute error function.
$\mathcal{E}_{\text{mse}}$ . . . . . Mean squared error function.
$\mathcal{E}_{\text{rmae}}$ . . . . . Relative mean absolute error function.

$\mathscr{G}$ . . . . . Similarity Graph.
G . . . . . Number of attribute groups.
γ . . . . . Regularization parameter for RR.

$\mathbf{K}$ . . . . . Kernel matrix over the graph.

$\mathbf{L}$ . . . . . Laplacian matrix.
$\mathscr{L}$ . . . . . Latent Space.
∇ . . . . . Laplacian operator.
λ . . . . . Eigenvalue of the probability matrix.
$\mathbf{\Lambda}$ . . . . . Eigenvalue matrix.

M . . . . . Dimension of the original data sample.
$\bar{M}$ . . . . . Reduced dimension of the embedding or projected space.
$\mathscr{M}$ . . . . . Original manifold.

N . . . . . Number of patterns of the original data sample.

$\mathbf{P}$ . . . . . Transition probability matrix of the Markov chain.
P . . . . . Meteorological variable: pressure.
p . . . . . Energy production.
P_{tr} . . . . . Number of training patterns.
ψ . . . . . Left eigenvectors of the probability.
φ . . . . . Right eigenvectors of the probability.
$\mathbf{\Phi}$ . . . . . Eigenvector matrix.
ϕ . . . . . Eigenvector of the symmetrized probability matrix.
ϑ . . . . . Eigenvectors of the covariance matrix.
$\mathbf{\Psi}$ . . . . . Diffusion coordinates.
Π . . . . . Stationary Distribution of the Markov chain.

ρ . . . . . Distance percentile.

$\mathscr{S}$ . . . . . Original data sample.
$\mathcal{S}$ . . . . . Dimensionality reduction function.
σ . . . . . Kernel width parameter.

T . . . . . Meteorological variable: temperature.
T . . . . . Diffusion step.

VAR . . . . . Variance explained by the Principal Components.
V . . . . . Meteorological variable: wind speed norm at surface level.
V_x . . . . . Meteorological variable: horizontal (x) component of wind speed at surface level.
V_y . . . . . Meteorological variable: vertical (y) component of wind speed at surface level.

$\mathbf{W}$ . . . . . SC adjacency matrix over the graph.

$\mathbf{y}$ . . . . . Target vector.
$\hat{\mathbf{y}}$ . . . . . Target predicted vector.

To Carlos.

Chapter 1

Introduction

1.1 Big Data and Machine Learning

The 21st century has started with an important fact looming on the horizon: the information explosion. The incredible improvements in technology that are taking place nowadays give rise to a huge quantity of digital information that is getting bigger every day. If we stop for a moment thinking about the quantity of information that we generate per day and the enormous amount of information we have access to, we will realize about the fascinating problem we are dealing with. Computers, mobile phones, dozens of different gadgets, GPS systems, multitude of different industrial sensors, electrical meters, weather-vanes, anemometers, scanners... all of them are generating tons of digital information every second. And this without talking about the billions of queries that Google received, the number of videos uploaded at Youtube, the amount of tweets, Facebook news or Instagram photos posted per minute around the world.

To properly collect, organize, summarize, analyze and synthesize this data to finally take a decision based on it, is a difficult challenge that companies have to deal with in this age. These different steps can be seen in Figure 1.1. And in this way is how it was born what scientists and computer engineers have called *Big Data*.

What exactly big data is? The answer will probably depend on whom you ask and his area of knowledge, as it is a very new term not established for good yet. Even though, we can define big data as "the capability to manage a huge volume of disparate data, at the right speed, and within

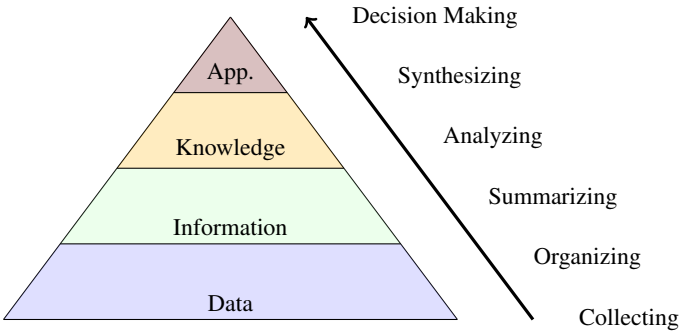


FIGURE 1.1: The big data pyramid: how to transform raw data into applicable information [Mohanty et al., 2013].

the right time frame to allow real-time analysis and reaction" [Hurwitz et al., 2013]. What this and other multiple definitions of big data have in common is what Laney [2001] called the *3Vs model*, which basically states the three main characteristics of big data:

- **Volume**: how much data is available.
- **Velocity**: how fast that data could be processed.
- **Variety**: how dissimilar the data is.

In any case, to apply this model in an excessively strict way could make us lose sight of the problem in question, which should be never underestimate. A problem with a small amount of disparate, complex data or a huge problem of very easy data can be also considered as big data problems. This brings to a fourth "V", namely **Veracity**, i.e., how accurate data is for predicting a value of interest [Hurwitz et al., 2013].

One difficulty that arises with this definition is that the meaning of these characteristics changes very quickly, as, for example if a huge amount of data several years ago was a question of gigabytes, nowadays we will be talking about exabytes. Another difficulty lies in the variety of the data, as structured, unstructured and multi-structured data can appear mixed in the same volume of information.

This takes to the big challenge in this kind of problems: how to manage data and how to extract information to make decisions. And at this point is where *Knowledge Discovery and Data Mining (KDD)* and *Machine Learning (ML)* can be introduced. These two areas are very closed and related, even sharing many methods, but they are different, particularly in their focus. Arthur L. Samuel, a pioneer of Artificial Intelligence research, defined in 1959 ML as the field of study that gives computers the ability to learn without being explicitly programmed [Simon, 2013], and it can be said that this area is focused on learning data from a well understood training set. KDD is the computational process of discovering patterns in large data sets involving methods

at the intersection of artificial intelligence, machine learning, statistics, and database systems [Soumen et al., 2006], and it can be said that is based on discovering unknown properties of the data.

In both interdisciplinary subfields of computer sciences we can distinguish two types of learning problems to deal with: *supervised learning*, where a category label or cost is provided for each pattern in a training set, and *unsupervised learning* or *clustering*, where no labels are provided and the system looks for natural groups (clusters) of the input data [Duda et al., 2001]. Big data problems could be of any of these types, but non-labeled data is more common when we have to face a huge volume of information.

A first way to face big data is trying to represent it, which can be done by selecting important features and defining a measure of similarity over them. In this context, a first approach is to use feature selection techniques, which try to highlight the most promising features in an automatic way. The problem when just selecting a subset of the original features is that some information can be lost. Dimensionality reduction by feature extraction is an interesting alternative to feature selection as it also yields a low dimensional representation that can, in general, help predictive models to get a better generalization, and moreover, preserve information from all the original input variables [Bengio et al., 2006]. We will see how beyond classical dimensionality methods like Principal Component Analysis (PCA), nonlinear spectral dimensionality reduction may perform better in more general scenarios. Notice that in this thesis we will focus on filter methods, which are those that look for a new representation based only on the data, whereas wrapper methods make use of predictive models to score feature subsets. This allows us to obtain general, model-independent representations than can be used for visualization and data analysis.

When the data is properly represented in a low dimensional space where it can be processed, clustering could be a useful technique. Clustering can be defined as the formal study of algorithms and methods for grouping, or classifying, objects. Each cluster will be comprised of a number of similar objects collected or grouped together [Jain and Dubes, 1988]. As Anil K. Jain claims in his talk "Clustering Big Data" [Jain, 2013], clustering can be considered as one of the key tools in big data problem, as it does not need labeled data, it does not need to know the number of instances of each group and it does not even need to know the number of groups. When it works properly, it is able to detect outliers or to measure similarity; and moreover, there exist several methods to estimate cluster quality. Although there is not a universally optimal clustering algorithm, the classical, simple κ-means algorithm is often a very good option but only if the natural distances in the space are Euclidean. We will see in this work how nonlinear spectral dimensionality reduction techniques can embed the data in a Euclidean space without losing information, being then possible to successfully apply κ-means to big data problems.

1.2 Classical Clustering and Dimensionality Reduction

In this thesis, we are going to principally focus on a specific subtype of big data: high dimensional data. As previously explained, dimensionality reduction and clustering are among the main tools to deal with this kind of big data problems; thus we are going to present here in detail the two principal, classical techniques in these areas: κ-means and PCA.

1.2.1 k-means

κ-means is the most popular, simplest and extensively used clustering algorithm. It was independently discovered in different scientific fields; see for instance Steinhaus [1956], Ball and Hall [1965], MacQueen [1967] and Lloyd [1982]. These different origins make it possible to justify κ-means from different points of view, but in this brief review we are going to follow a similar approach to that in Jain [2010] or Duda et al. [2001].

Let $\mathscr{S} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ be a data set of N M-dimensional points that we want to group in κ clusters $\mathscr{C} = \{\mathscr{C}_j\}_{j=1}^{\kappa}$. The κ-means algorithm tries to minimize the intra-cluster distances while maximizing the inter-cluster separations. For this to be done, we define an objective function that summarizes these requirements, trying to find the partition that minimize the ℓ_2-distance between the points inside each cluster and its centroid:

$$\mathcal{J}(\mathscr{C}) = \sum_{k=1}^{\kappa} \sum_{\mathbf{x}^{(i)} \in \mathscr{C}_k} \|\mathbf{x}^{(i)} - \mu_k\|^2, \tag{1.1}$$

where μ_k are the centroids of each cluster $\mathscr{C}_k$.

This algorithm follows an iterative refinement technique, and departing from an initial set of centroids, κ-means assigns points to each nearest centroid and computes new centroids as the mean of each cluster. This procedure will be repeated until no changes occur in the centroids. It can be proved that these steps minimize the objective function in Equation (1.1). The κ-means algorithm is summarized in Algorithm 1.1.

The decrease of $\mathcal{J}$ is assured because each step reduces its value, but it is possible that it converges to a suboptimal solution. In fact, it will only converge to the optimal solution if the clusters are well separated [Meilă, 2006]. It is often useful to normalize the data, which is usually done by standardizing variables such that they have zero-mean and unit standard deviation. A good initialization of $\mu = \{\mu_i\}_{i=1}^{\kappa}$ is to take a random subset of κ points from the sample data [Bishop, 2006].

When we receive a new data point, we do not have to repeat the whole algorithm to assign it to a cluster. Once we have calculated the clusters' centroids, we only need to find the nearest centroid

ALGORITHM 1.1: κ-means Algorithm.

Input: $\mathscr{S} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$, the original data set ; μ, the initial centroids ; Parameters: κ ;

Output: $\mathscr{C} = \{\mathscr{C}_j\}_{j=1}^{\kappa}$, the data partition ; $\mu = \{\mu_1, \cdots, \mu_\kappa\}$ the final centroids ;

1: **repeat**
2: Assign to $\mathscr{C}_k$ the $\mathbf{x}^{(i)}$ nearest to μ_k ;
3: **for** $k = 1, \cdots, \kappa$ **do**
4: $\mu_k = \text{mean}(\mathbf{x}^{(i)} \in \mathscr{C}_k)$;
5: **end for**
6: **until** $\mu = \{\mu_1, \cdots, \mu_\kappa\}$ do not change

to our new point. This means that we should compute the distance between the new point and the different clusters' centroids and assign the new example to the cluster corresponding to the minimum distance.

An example of the behavior of this method is shown in Figure 1.2, where we depict the clusters obtained when applying κ-means over a sample formed by random points that follow three Gaussian processes of different mean and deviation. In the image the κ-means centroids are also depicted in gray, and the original mean of each Gaussian in black. It can be seen how the algorithm approximates properly the centroids to the real means of the target groups, and it is able to properly separate the three different distributions, although it mixes points from different Gaussians when their distributions overlap, as it only makes polyhedral cuts.

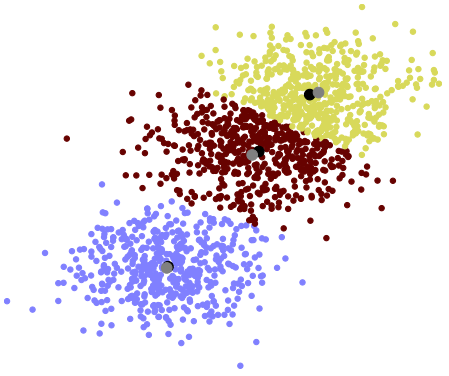


FIGURE 1.2: A κ-means example.

κ-means, as we have just seen, is a very simple algorithm but it often performs very well, being computationally relatively fast even with a large number of variables involved. Nevertheless, it presents some disadvantages, as for example, that we should prefix the number of clusters κ. This algorithm can also be costly because it is necessary to compute the Euclidean distance between every centroid and every data point, although this problem can be alleviated as shown in Bishop [2006]. Moreover, it does not work well with non-globular clusters. The use of the Euclidean distance as dissimilarity measure limits the type of data variables that can be considered and can make the cluster determination non-robust to outliers. This last disadvantage is the main problem of κ-means as it is unusual that the natural distance in a data set is Euclidean.

1.2.2 Principal Component Analysis

Working in low dimensions is always easier and has many computational advantages. Consequently, dimensionality reduction is usually desirable when we work with learning problems and probably mandatory when their dimension is high.

Let $\mathscr{S} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$ with $\mathbf{x}^{(i)} \in \mathbb{R}^M$, be our initial sample. The general dimensionality reduction problem looks for another representation in set $\{\hat{\mathbf{x}}^{(1)}, \dots, \hat{\mathbf{x}}^{(N)}\}$, with $\hat{\mathbf{x}}^{(i)} \in \mathbb{R}^{\bar{M}}$, where $\bar{M} \ll M$. In this way, the points $\hat{\mathbf{x}}^{(i)}$ represent the original points $\mathbf{x}^{(i)}$ in our new data space.

There exist several classical dimensionality reduction methods like Principal Component Analysis (PCA) [Jolliffe, 2002], Multidimensional Scaling (MDS) [Cox and Cox, 2000], Isomap [Tenenbaum et al., 2000] or Self-Organizing Maps (SOM) [Kohonen, 1988] that obtain a low-dimensional embedding where some properties of the original data are preserved.

In this subsection we are going to explain in more detail one of the most widely used dimensionality reduction methods: *Principal Component Analysis (PCA)*. It is a method used to compress the information of a data set, based on the idea of finding a new subset with the most relevant components or factors from the original features in the sense that they retain most of the original sample's variance. In this way, we obtain a dimensionality reduction and a new expression for the data. More concretely, we are going to transform the original features into some other features, called *Principal Components*, by applying linear combinations of the original ones. Thus, while reducing the original dimension PCA tries to preserve as much as possible the data variance in the original high dimensional space.

Let $\mathbf{x}$ be a M-dimensional, 0-mean and 1-std normalized vector and $\{\vartheta_1, \vartheta_2, \dots, \vartheta_M\}$ an orthonormal basis, i.e,

$$\vartheta_i \cdot \vartheta_j = \begin{cases} 0 & i \neq j \\ 1 & i = j. \end{cases}$$

We can express $\mathbf{x}$ as a linear combination of these vectors as

$$\mathbf{x} = \sum_{i=1}^{M} y_i \vartheta_i,$$

where y_i are the coefficients of the linear combination, that can be computed as

$$y_j = \mathbf{x}^\top \vartheta_j \quad \forall j.$$

If we want to represent $\mathbf{x}$ only with the first $\bar{M}$ ($\bar{M} \ll M$) vectors in the basis and they are properly ordered, we can ignore the coefficients $[y_{\bar{M}+1}, \ldots, y_M]^\top$, obtaining the following expression:

$$\hat{\mathbf{x}} = \sum_{i=1}^{\bar{M}} y_i \vartheta_i.$$

We can compute the reconstruction error as a mean squared error of this approximation,

$$\begin{aligned}
\mathcal{E}_{\text{mse}}^2 &= \mathrm{E}\left[(\mathbf{x} - \hat{\mathbf{x}})^2\right] \\
&= \mathrm{E}\left[\left(\sum_{i=\bar{M}+1}^{M} y_i \vartheta_i\right)^2\right] \\
&= \mathrm{E}\left[\sum_{i=\bar{M}+1}^{N} \sum_{j=\bar{M}+1}^{N} y_i y_j \vartheta_i^\top \vartheta_j\right] \\
&= \sum_{i=\bar{M}+1}^{N} \mathrm{E}\left[(y_i)^2\right] \\
&= \sum_{i=\bar{M}+1}^{N} \vartheta_i^\top \, \mathrm{E}[\mathbf{x}\mathbf{x}^\top] \vartheta_i \\
&= \sum_{i=\bar{M}+1}^{N} \vartheta_i^\top \mathbf{C} \vartheta_i,
\end{aligned}$$

where $\mathbf{C}$ is the sample's covariance matrix, which coincides with the correlation matrix if the data has been previously standardized.

The PCA algorithm looks for the orthonormal basis ϑ that minimizes this reconstruction error, so it has to solve the minimization problem

$$\min_{\vartheta} \{\mathcal{E}_{\text{mse}}^2\} \quad s.t.\ \vartheta \text{ is an orthonormal basis.}$$

We can solve it analytically achieving a closed solution, as shown in Bishop [2006], given by the eigenvectors of the covariance matrix $\mathbf{C}$. This implies that the mean square error can be expressed as $\mathcal{E}_{\text{mse}} = \sum_{i=\bar{M}+1}^{N} \lambda_i$, where λ_i are the corresponding eigenvalues to the discarded eigenvectors, and therefore the Principal Components are given by the $\bar{M}$ eigenvectors that correspond to the highest eigenvalues of the covariance matrix. An example of how this method behaves is shown in Figure 1.3. In this image, we can see how this technique rotates the axis of the original data, projecting the variables over the axis with a higher variance. We present a summary of this method in Algorithm 1.2.

PCA is a classical dimensionality reduction algorithm with many advantages, as it applies a data transformation to the sample data obtaining new features that are uncorrelated in the projected space. Because of this it is a useful technique for feature extraction, for outlier detection or

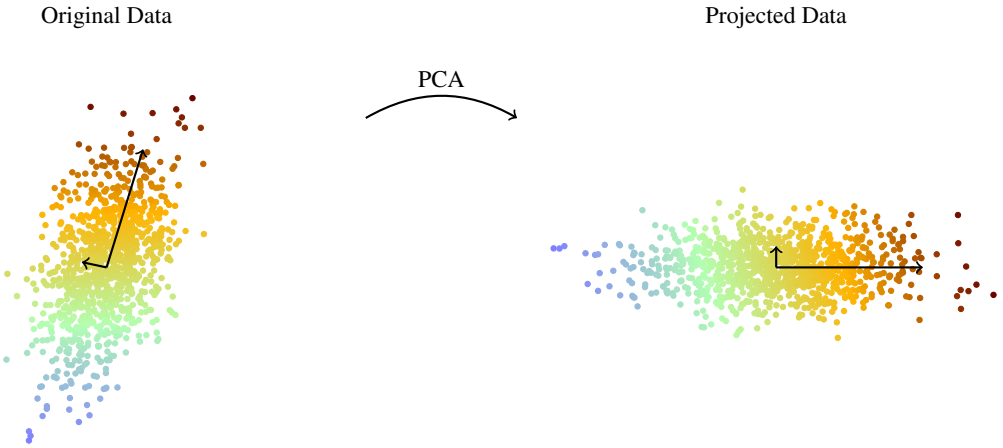


FIGURE 1.3: A PCA example.

ALGORITHM 1.2: PCA Algorithm.

Input: $\mathscr{S} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\} \in \mathbb{R}^M$;

Output: PC, Principal Components computed ;

1: $\tilde{\mathscr{S}} = \frac{\mathscr{S}-\mu}{\sigma}$, with μ $\mathscr{S}$-mean and σ its std ; ▶ *$\mathscr{S}$ Normalization.*

2: $\mathbf{C} = \mathrm{E}[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^\top]$; ▶ *Covariance Matrix.*

3: $\lambda = \{\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{\bar{M}}\}$ and $\vartheta = \{\vartheta_i\}$; ▶ *$\mathbf{C}$ eigendecomposition.*

4: $PC = \{PC_i\}_{i=1}^{\bar{M}} = \{\vartheta_i^\top \tilde{\mathbf{x}}^{(i)}\}_{i=1}^{\bar{M}}$; ▶ *$\bar{M}$ Principal Components.*

clustering, and it has the good property of working well when the features present a high correlation, since a few factors will explain a high part of the total variability. But it presents some important disadvantages. This method only rotates the coordinates, changing the data axis in the maximum variance direction, and there does not exist any guarantee of having good classification or predictive information in the maximum variance axis. And it only takes into account linear relations between the features, so it does not work properly for more complicated feature relationships. This is a common potential disadvantage in classical dimensionality reduction and clustering methods, as they do not consider explicitly or implicitly the structure of the differentiable manifold in which the data probably lie. In practice, this fact gives rise to difficulties such as the misclassification of non-globular clusters when applying κ-means over the embedding obtained.

1.2.3 Manifold Learning

Manifold learning algorithms appear as an alternative to classical dimensionality reduction methods, often having the property of "unfolding" the manifold where the original data is embedded. These techniques can be used over unsupervised or supervised problems, discovering

high-density low-dimensional surfaces in the first case, or used as a preprocessing step in the second one, being very useful, for example, for data analysis and visualization [Bengio et al., 2006].

Their goal is to achieve a new representation that captures the structure of the data in a few dimensions while preserving the original local information. For this to be done, most of these methods rely on the spectral analysis of a similarity matrix of a graph previously constructed over the original data. Another important characteristic of these methods is that they may arrive at a new space where the Euclidean distance between embedded points corresponds with the original information preserved. This characteristic makes possible to apply κ-means over the embedded data in a principled way and to obtain good results.

In this thesis we are going to work in this framework, presenting and studying some of the most important nonlinear spectral dimensionality reduction techniques such as Locally Linear Embedding (LLE) [Roweis and Saul, 2000], Laplacian Eigenmaps (LE) [Belkin and Nyogi, 2003], Spectral Clustering (SC) [Luxburg, 2007] or Diffusion Maps (DM) [Coifman and Lafon, 2006a].

1.3 Thesis Contributions and Structure

1.3.1 Contributions

The thesis contributions have both an algorithmic and an application-oriented nature. From an algorithmic point of view, we introduce first an adaptation of the Anisotropic Diffusion (AD) method that exploits the approximation of the latent variable distance by a local Mahalanobis one and combines it with a κ-Nearest Neighbors (κ-NN) search to build local models in a principled way without the need of embedding the sample data. We also present the Auto-adaptative Laplacian Pyramids (ALP) method, a modification of the Laplacian Pyramids (LP) algorithm for function interpolation that combines the LP training phase with an estimate of the Leave One Out CV (LOOCV) error, and makes possible to directly define during training a criterion to estimate the optimal stopping point of the LP iterations. This approach lightens the computational cost of LP training and helps to avoid overfitting, as the automatic stopping point corresponds to a minimum of the estimated LOOCV error. While the ALP method can be applied on a wide variety of function extension problems, our main goal here has been its use as a tool to extend DM and AD embeddings to out-of-sample patterns. This is a key issue when applying DM and AD in big data contexts and in this vein, we also propose several metrics to compare different out-of-sample methods.

From an application point of view, the DM procedure, and the proposed AD extension have been applied to several important problems in the prediction of wind and solar energy, and the body location of CT scan images. In some of them, the just mentioned ALP method has been applied to compute the embedding coordinates of new test patterns. More concretely, we have exploited DM as a pattern compression and visualization tool on large dimension wind energy related meteorological data, and also used it to build data clusters over which we build local wind energy models. In turn, we have applied AD to two very important practical problems. The first one is the prediction of the location of CT scan images in the human body, where our AD proposal improves on the current state-of-the-art methods. The second one is the detection of incoming wind ramps. The very rapid spread of wind energy and its increasing impact on the management of electricity transmission systems makes the accurate and efficient prediction of extreme events such as wind ramps an important issue for the Transmission System Operator (TSO) that, among other responsibilities, has to maintain the stability of the electrical system. In this thesis a complete framework, based on AD, to deal with this phenomenon is proposed and we have built a procedure to detect ramps that presents high precision and recall. Finally, we have compared different out-of-sample methods for the extension of DM coordinates on two large dimensionality problems, the time compression of meteorological data and the analysis of meteorological variables, relevant in daily radiation forecasts. Applying the previously mentioned metrics, we show that ALP compares favorably with other LP approaches and the Nyström method.

We want to point out that we have put together all the diffusion methods applied in this thesis in a Matlab toolbox that is publicly available at the software website of the Machine Learning Group at Universidad Autónoma de Madrid [GAA, 2014].

Finally, we mention that after reviewing the main manifold learning methods, the thesis presents in a complete systematic, compact and self-contained way the general theory of the DM and AD diffusion methods introduced by Ronald R. Coifman and his school at Yale University. While, of course, not original, this review might also be considered as a scholarly valuable contribution.

1.3.2 Structure

This thesis is organized in six chapters, and a brief summary of their contents is shown next.

Chapter 1: Introduction. In this chapter we briefly motivate manifold learning methods from the point of view of having to deal with high dimensional problems in big data and then we present and analyze the well-known κ-means clustering algorithm and the classical PCA method for dimensionality reduction.

Chapter 2: Spectral Dimensionality Reduction. In this chapter we review in some detail and following a common perspective the most common nonlinear spectral dimensionality reduction methods, namely LE, LLE and finally SC, which can be seen as the first step towards DM.

Chapter 3: Diffusion Maps. In this chapter DM, that assumes that data lie in a manifold whose metric is related to a diffusion process, is thoroughly described, including its motivation, theoretical justification and parameterization, as well as an extension for applying it to heterogeneous data. Moreover, we apply this method to two different problems: the dimensionality reduction and compact description of meteorological data and the clustering of wind power data with a goal of building local prediction models.

Chapter 4: Anisotropic Diffusion. This chapter reviews the algorithm of AD, another spectral dimensionality reduction technique strongly related with DM but where the underlying hypothesis is that observable data correspond to latent variables that follow an Itô process. Moreover, a methodology to combine this method with the classical κ-NN model in his regression version is proposed and applied to two real-world problems: the prediction of CT scan images location and the detection of wind power ramps.

Chapter 5: Out-of-sample Methods. This chapter analyzes the problem of extending the embedding provided by DM over new, unseen data. In particular, two state-of-the-art approaches are described and compared, the Nyström method and the LP algorithm. Furthermore, a much more efficient extension of the latter, namely ALP, is introduced. The three approaches are compared over two different real meteorological-based data sets.

Chapter 6: Conclusions. In this chapter we present some general conclusions that can be derived from this work and provide some possible lines of future work.

Capítulo 6: Conclusiones. This chapter is just a Spanish version of Chapter 6.

Appendix A: Relation of Publications. This appendix includes a relation of the publications corresponding to this thesis, with a brief summary of each one.

Chapter 2

Spectral Dimensionality Reduction

2.1 Introduction

Spectral dimensionality reduction methods are dimensionality reduction methods based on performing an eigendecomposition of a similarity matrix defined over the sample data, that we will denote by $\mathscr{S} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ with $\mathbf{x}^{(i)} \in \mathbb{R}^M$. This definition fits also for classical dimensionality reduction methods, such as Principal Component Analysis (PCA), but in this chapter we are going to focus our attention over those methods that look for the intrinsic geometry of the data using nonlinear mappings.

The general dimensionality reduction problem aims to find a new representation $\{\hat{\mathbf{x}}^{(1)}, \ldots, \hat{\mathbf{x}}^{(N)}\}$ with $\hat{\mathbf{x}}^{(i)} \in \mathbb{R}^{\bar{M}}$, where $\bar{M} \ll M$ represent the original points $\mathbf{x}^{(i)}$ in the new data space. For manifold learning tasks, special attention is paid to the case where the patterns $\mathbf{x}^{(i)}$ are in a differentiable manifold $\mathscr{M} \subset \mathbb{R}^M$ of dimension $\bar{M}$.

In this chapter we are going to review some spectral dimensionality reduction methods, starting with Laplacian Eigenmaps (LE) in Section 2.2 where the method and its theoretical justification are presented, followed by a brief review of Locally Linear Embedding (LLE) as another example of geometric neighborhood information-based method in Section 2.3 and, finally, in Section 2.4 the Spectral Clustering (SC) technique and its different algorithms are presented, accompanied by a complete theoretical justification from different points of view.

2.2 Laplacian Eigenmaps

The Laplacian Eigenmaps (LE) method is a very successful, recently proposed procedure to reduce dimensionality for semi-supervised learning [Belkin and Nyogi, 2003] while preserving the local information. For this purpose, and as in all the spectral dimensionality reduction methods that we will study in this thesis, it will organize the sample data $\mathscr{S}$ as a weighted graph over which it can be computed the Laplacian matrix. This matrix captures the local geometry information presented in the original data, and allows building an embedding that preserves non-linear relations between points. More concretely, LE builds a weighted graph of N nodes from a set of points $\mathscr{S} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$ with $\mathbf{x}^{(i)} \in \mathbb{R}^M$, defining links only between neighbors. The aim is to obtain an embedded map by computing the eigenvectors of the graph Laplacian. Let us describe the formal aspects of this algorithm.

Step 1: Constructing the adjacency graph. We are going to define the adjacency graph as

$$\mathscr{G} = (\mathscr{V} = \{\mathbf{x}^{(i)}\}, \mathscr{E}),$$

where $\mathscr{E}$ are the edges of the graph and $\mathscr{V}$ the vertices. The vertices will coincide with the points of our original sample, and the edges will be established between two points when they are near, i.e. $(i,j) \in \mathscr{E}$ if $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ are close to each other in some sense. To decide when two points are neighbors we can use different distance-based methods as ϵ-*neighborhood graphs*, where only the points separated by a distance smaller than ϵ are connected, κ-*Nearest Neighbors graphs*, that only connects the κ closest points to a given $\mathbf{x}^{(j)}$ or *fully connected graphs*, that defines the connections between nodes according to the weights described by a similarity function [Belkin and Nyogi, 2003].

Step 2: Choosing the weights. The graph constructed above will be always a weighted graph, with weights w_{ij} defined by one of these two options:

- The *Simple-minded* method, where the only possible values for our weights are 0 or 1, i.e.

$$w_{ij} = \begin{cases} 1 & \text{if } (i,j) \text{ are connected,} \\ 0 & \text{if } (i,j) \text{ are not connected.} \end{cases}$$

  The obvious advantage is that we do not have to fix any parameter, but it gives us less information about the local structure of the data.

- The *Heat Kernel* method with parameter $t \in \mathbb{R}$. In this case, we will set up the weights in the following way

$$w_{ij} = \begin{cases} e^{-\frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2}{4t}} & \text{if } (i,j) \text{ are connected,} \\ 0 & \text{if } (i,j) \text{ are not connected.} \end{cases}$$

  With this method, we can expect to obtain more information about the relationships inside the data set.

Step 3: Computing eigenmaps. Once we have a connected graph $\mathscr{G}$ constructed following steps 1 and 2, we must define the map for the embedded coordinates. For this purpose, we work with the unnormalized graph Laplacian $\mathbf{L}$ of $\mathscr{G}$.

Definition 2.1 (Degree Matrix)**.** We define the *Degree Matrix* $\mathbf{D}$ of the similarity matrix $\mathbf{W}$ as the diagonal matrix with entries

$$d_i = \sum_{j=1}^{N} w_{ij}.$$

Definition 2.2 (Unnormalized Graph Laplacian)**.** We define the *Unnormalized Graph Laplacian* $\mathbf{L}$ as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}.$$

To find the new coordinates in the embedded space we must compute the eigenvalues λ_i and eigenvectors ϑ_i of $\mathbf{L}$ via the generalized spectral problem $\mathbf{L}\vartheta = \lambda\mathbf{D}\vartheta$ [Belkin and Nyogi, 2003]. $\mathbf{L}$ has the special characteristic of being a positive semidefinite matrix (see below Lemma 2.1), so its eigenvalues are always positive ($\lambda_i \geqslant 0$).

Lemma 2.1 ($\mathbf{L}$ Positive Semidefinite Matrix)**.** *The unnormalized graph Laplacian* $\mathbf{L}$ *is a positive semidefinite matrix.*

Proof. To prove that $\mathbf{L}$ is positive semidefinite we check that $\vartheta^\top \mathbf{L}\vartheta \geqslant 0$. We have

$$\begin{aligned}
\vartheta^\top \mathbf{L}\vartheta &= \vartheta^\top \mathbf{D}\vartheta - \vartheta^\top \mathbf{W}\vartheta \\
&= \sum_{i=1}^{N} d_i \vartheta_i^2 - \sum_{i,j=1}^{N} \vartheta_i \vartheta_j w_{ij} \\
&= \frac{1}{2}\left(\sum_{i=1}^{N} d_i \vartheta_i^2 - 2 \sum_{i,j=1}^{N} w_{ij}\vartheta_i\vartheta_j + \sum_{j=1}^{N} d_j \vartheta_j^2 \right) \\
&= \frac{1}{2}\left(\sum_{i,j=1}^{N} w_{ij} \vartheta_i^2 - 2 \sum_{i,j=1}^{N} w_{ij}\vartheta_i\vartheta_j + \sum_{i,j=1}^{N} w_{ji} \vartheta_j^2 \right) \\
&= \frac{1}{2} \sum_{i,j=1}^{N} w_{ij}(\vartheta_i - \vartheta_j)^2 \geqslant 0,
\end{aligned}$$

where we have applied Definition 2.1 and Definition 2.2. □

Under these conditions, we can solve the spectral problem obtaining the solutions $\vartheta_0, \vartheta_1, \ldots, \vartheta_{\bar{M}}, \ldots, \vartheta_{N-1}$ ordered by their corresponding eigenvalues in an increasing way

$$0 = \lambda_0 < \lambda_1 < \ldots < \lambda_{N-1}.$$

Notice that $\lambda_0 = 0$ is always a trivial eigenvalue as, thanks to the normalization, our matrix satisfies:

$$\begin{pmatrix} d_1 - w_{11} & \ldots & -w_{1n} \\ \vdots & & \\ -w_{n1} & \ldots & d_n - w_{nn} \end{pmatrix} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = 0 \begin{pmatrix} 1 & \ldots & 1 \end{pmatrix} = \begin{pmatrix} 0 & \ldots & 0 \end{pmatrix}.$$

We are not going to consider the trivial solution associated to this $\lambda_0 = 0$, because the eigenfunction $\vartheta_0 : \mathbf{x}^{(i)} \to (1, \ldots, 1)$ collapses all the elements of each point onto the real number 1. Thus, it gives in a trivial way a projection with a minimum distance between points (as it is zero) but we lose all information.

The embedding in a $\bar{M}$-dimensional space is obtained then with the first $\bar{M}$ eigenvectors, without taking into account ϑ_0 [Belkin and Nyogi, 2003],

$$\vartheta : \mathbf{x}^{(i)} \to (\vartheta_1(\mathbf{x}^{(i)}), \ldots, \vartheta_{\bar{M}}(\mathbf{x}^{(i)})).$$

2.2.1 Optimal Embedding over a Graph

We have presented the LE method as a good technique for dimensionality reduction because it may preserve the local information, but we had not given any reasoning to prove this statement.

In this subsection we will show how the above embedding $\mathbf{Y} : \mathscr{G} \to \mathbb{R}^{\bar{M}}$ minimizes a reconstruction error. We assume that $\mathscr{G}$ is connected and suppose that $\bar{M} = 1$ to simplify notation and explanations. A good map will minimize the objective function defined by the following error criterion:

$$\mathcal{J}(\mathbf{y}) = \frac{1}{2}\sum_{i,j}(y_i - y_j)^2 w_{ij} \geqslant 0. \tag{2.1}$$

Applying Definition 2.1 and Definition 2.2, Equation (2.1) can be rewritten as follows:

$$\begin{aligned}
\frac{1}{2}\sum_{i,j}(y_i - y_j)^2 w_{ij} &= \frac{1}{2}\sum_{i,j}(y_i^2 + y_j^2 - 2y_i y_j) w_{ij} \\
&= \frac{1}{2}\left[\sum_i d_i y_i^2 + \sum_j d_j y_j^2 - 2\sum_{i,j} w_{ij}(y_i y_j)\right] \\
&= \frac{1}{2}\left[\mathbf{y}^\top \mathbf{D}\mathbf{y} + \mathbf{y}^\top \mathbf{D}\mathbf{y} - 2\mathbf{y}^\top \mathbf{W}\mathbf{y}\right] \\
&= \frac{1}{2}\left[2\mathbf{y}^\top \mathbf{D}\mathbf{y} - 2\mathbf{y}^\top \mathbf{W}\mathbf{y}\right] \\
&= \frac{1}{2}\left[2\mathbf{y}^\top (\mathbf{D} - \mathbf{W})\mathbf{y}\right] \\
&= \mathbf{y}^\top \mathbf{L}\mathbf{y}.
\end{aligned} \tag{2.2}$$

In other words, to minimize Equation (2.1) is the same as solving the matrix minimization problem

$$\begin{aligned}
&\arg\min_{\mathbf{y}}\{\mathbf{y}^\top \mathbf{L}\mathbf{y}\} \\
&s.t.\ \mathbf{y}^\top \mathbf{D}\mathbf{y} = 1,
\end{aligned} \tag{2.3}$$

where we have added a restriction to remove any arbitrary scaling factor and then avoid degenerated solutions. Normally, the restriction $\|\mathbf{y}\|^2 = 1$ is applied, but in this case the matrix $\mathbf{D}$ seems to be the natural measure over the graph as it makes reference to the vertices relationship (a big value of d_i means that the vertex $\mathbf{x}^{(i)}$ is very connected and, in consequence, it is more important).

To solve Problem (2.3) we can use Lagrange multipliers by rewriting the equations as follows

$$\begin{aligned}
\phi(\mathbf{y}) &= \mathbf{y}^\top \mathbf{L}\mathbf{y} - \lambda(\mathbf{y}^\top \mathbf{D}\mathbf{y} - 1), \\
\nabla\phi(\mathbf{y}) &= \mathbf{L}\mathbf{y} - \lambda\mathbf{D}\mathbf{y} = 0,
\end{aligned}$$

thus the solution is the eigenvector $\mathbf{y}$ corresponding to the minimum eigenvalue λ of $\mathbf{L}$ that satisfies

$$\mathbf{L}\mathbf{y} = \lambda\mathbf{D}\mathbf{y}.$$

As explained at the beginning of Section 2.2, the vector $\mathbf{y} = \mathbf{1} = (1, \dots, 1)$ corresponds to an eigenvalue 0, and we are not interested in this trivial solution. Because of this, we eliminate this possibility adding another restriction to the minimization problem, forcing to find always a solution orthogonal to the trivial one, and the problem becomes

$$\begin{aligned} &\arg\min_{\mathbf{y}}\{\mathbf{y}^\top \mathbf{L}\mathbf{y}\} \\ s.t. &\begin{cases} \mathbf{y}^\top \mathbf{D}\mathbf{y} = 1 \\ \mathbf{y}^\top \mathbf{D}\mathbf{1} = 0. \end{cases} \end{aligned}$$

In this case, the solution is given by the eigenvector corresponding to the smallest non-zero eigenvalue, and thus, this problem arrives to the same solution that we had when solving LE. We can then conclude that LE is equivalent to this minimization problem, and thus it is a good method for preserving local information when $\bar{m} = 1$.

Let us now generalize this explanation to a $\bar{m} > 1$ dimensional embedding. In this case the embedding is given by the $n \times \bar{m}$ matrix $\mathbf{Y} = [\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(\bar{m})}]$, and we have to minimize the function

$$\mathcal{J}(\mathbf{Y}) = \sum_{ij} \|\mathbf{y}^{(i)} - \mathbf{y}^{(j)}\|^2 w_{ij},$$

that, following the same reasoning that in Equation (2.2), is equivalent to $\text{Tr}(\mathbf{Y}^\top \mathbf{L}\mathbf{Y})$:

$$\begin{aligned} \sum_{ij} \|\mathbf{y}^{(i)} - \mathbf{y}^{(j)}\|^2 w_{ij} &= \sum_{ij} \|((y_1^{(i)} - y_1^{(j)}), \dots, (y_{\bar{m}}^{(i)} - y_{\bar{m}}^{(j)}))\|^2 w_{ij} \\ &= \sum_{ij} \left((y_1^{(i)} - y_1^{(j)})^2 + \dots + (y_{\bar{m}}^{(i)} - y_{\bar{m}}^{(j)})^2\right) w_{ij} \\ &= \sum_{ij} (y_1^{(i)} - y_1^{(j)})^2 w_{ij} + \dots + \sum_{ij} (y_{\bar{m}}^{(i)} - y_{\bar{m}}^{(j)})^2 w_{ij} \\ &= \mathbf{y}_1^\top \mathbf{L}\mathbf{y}_1 + \dots + \mathbf{y}_{\bar{m}}^\top \mathbf{L}\mathbf{y}_{\bar{m}} \\ &= \text{Tr}(\mathbf{Y}^\top \mathbf{L}\mathbf{Y}). \end{aligned}$$

Therefore, the problem we have to solve can be written as

$$\begin{aligned} &\arg\min_{\mathbf{Y}}\{Tr(\mathbf{Y}^\top \mathbf{L}\mathbf{Y})\} \\ &s.t.\ \mathbf{Y}^\top \mathbf{D}\mathbf{Y} = 1, \end{aligned} \tag{2.4}$$

where we have considered a normalizing restriction similar to that of the $\bar{m} = 1$ case. Now, if we are interested in avoiding a collapse onto a $\bar{m}$-dimensional subspace, we should also add orthogonality restrictions, as done before [Belkin and Nyogi, 2003].

The solution to Problem (2.4) will be given by the eigenvector matrix corresponding to the lowest eigenvalues of the generalized spectral problem $\mathbf{L}\mathbf{Y} = \lambda \mathbf{D}\mathbf{Y}$, that we can achieve by solving

the Lagrange multipliers problem, as we have just explained for $\bar{M} = 1$ [Belkin and Nyogi, 2003]. We can then conclude that the general LE algorithm is a good method for embedding a sample preserving its local information.

2.2.2 The Laplace Beltrami Operator

As we have explained in the motivation to this chapter, we are interested in the case where the data lies in a smooth, compact, $\bar{M}$-dimensional manifold $\mathscr{M}$ embedded in the original space $\mathscr{M} \subset \mathbb{R}^M$. In this section we are going to briefly introduced how to work with the Laplace Beltrami operator, which is the equivalent on a manifold to the Laplacian of a graph, following the discussion on Belkin and Nyogi [2003].

We start by studying a map $f : \mathscr{M} \to \mathbb{R}$, from the manifold to the real line, where f should be at least twice differentiable, with the objective of sending nearby points in $\mathscr{M}$ to nearby points in $\mathbb{R}$. To formalize it, we think about two neighbor points $\mathbf{x}, \mathbf{z} \in \mathscr{M}$ and we study their difference in the new space $|f(\mathbf{z}) - f(\mathbf{x})|$. For this purpose, we consider a geodesic curve $\mathcal{C}$ parameterized by length with origin in $\mathbf{x}$, i.e.,

$$\begin{aligned} r &= d_{\mathscr{M}}(\mathbf{x}, \mathbf{z}), \\ \mathbf{z} &= \mathcal{C}(r), \\ \mathbf{x} &= \mathcal{C}(0), \\ f(\mathcal{C}(t)) &= g(t). \end{aligned}$$

With this notation, and since $f(\mathbf{x}) = f(\mathcal{C}(0)) = g(0)$ and $f(\mathbf{z}) = f(\mathcal{C}(r)) = g(r)$, we can rewrite the difference we are interested in as

$$\begin{aligned} f(\mathbf{z}) - f(\mathbf{x}) &= g(r) - g(0) \\ &= \int_0^r g'(t)dt \\ &= \int_0^r \nabla f(\mathcal{C}(t)) \cdot \mathcal{C}'(t)dt. \end{aligned}$$

Taking absolute values and using the Schwarz inequality, we arrive at

$$\begin{aligned}
|f(\mathbf{z}) - f(\mathbf{x})| &\leqslant \int_0^r \|\nabla f(\mathcal{C}(t))\|\|\mathcal{C}'(t)\|dt \\
&= \int_0^r \|\nabla f(\mathcal{C}(t))\|dt && (2.5)\\
&= \int_0^r \|\nabla f(\mathbf{x})\|dt + o(r) && (2.6)\\
&\leqslant r\|\nabla f(\mathbf{x})\| + o(r) && (2.7)\\
&= \|\mathbf{z} - \mathbf{x}\|\|\nabla f(\mathbf{x})\| + o(\|\mathbf{z} - \mathbf{x}\|). && (2.8)
\end{aligned}$$

In Equation (2.5) we have used that the geodesic curve $\mathcal{C}$ is parameterized by length, which means $\|\mathcal{C}'(t)\| = 1$. In the second equality, Equation (2.6), we used Taylor's approximation that tells us

$$\|\nabla f(\mathcal{C}(t))\| = \|\nabla f(\mathbf{x})\| + O(t).$$

And for the last equality, Equation (2.8), we use the distance definition over the manifold

$$d_{\mathscr{M}}(\mathbf{x}, \mathbf{z}) = r = \|\mathbf{z} - \mathbf{x}\| + o(\|\mathbf{z} - \mathbf{x}\|).$$

Now that we know how to measure distances over the manifold and how to relate them in the new space, we look for the map that better preserves local information. As we have done in the case over a graph (Section 2.2.1), we have to solve the minimization problem in terms of the reconstruction error. Note that, in this case, $\|\nabla f(\mathbf{x})\|$ gives a measure of the distortion between nearby points introduced by f (see Equation (2.7)); thus, we use it to define a criterion function. Consequently, the minimization problem we have to solve is given by

$$\begin{aligned}
&\min_f \{\int_{\mathscr{M}} \|\nabla f(\mathbf{x})\|^2\} \\
&s.t \parallel f\parallel_{\mathbf{L}^2(\mathscr{M})} = 1,
\end{aligned} \tag{2.9}$$

where the defined constrain removes scaling effects when working with unitary vectors.

This problem is closely related to the graph Laplacian, as we have

$$\int_{\mathscr{M}} \|\nabla f\|^2 = \int_{\mathscr{M}} \mathbf{L}(f)f.$$

In fact, by definition, $\mathbf{L}f = -div\nabla(f)$ and applying this equality and the Gauss's Divergence Theorem, we can see that

$$\begin{aligned}
\int_{\mathscr{M}} < \nabla f, \nabla f > &= -\int_{\mathscr{M}} div(\nabla(f))f \\
&= -\int_{\mathscr{M}} \mathbf{L}(f)f.
\end{aligned}$$

Therefore, to solve Problem (2.9) is equivalent to solve the minimization problem

$$\begin{aligned} &\min_{f}\{-\int_{\mathscr{M}} \mathbf{L}(f)f\} \\ &s.t\ \|f\|_{\mathbf{L}^2(\mathscr{M})} = 1, \end{aligned} \tag{2.10}$$

where we recall that $\mathbf{L}$ is a positive semidefinite operator, so a minimum of Problem (2.10) has to be given by an eigenfunction of $\mathbf{L}$ [Belkin and Nyogi, 2003].

As occurred in the graph case (Section 2.2.1), the optimal embedding will be formed by the first $\bar{M}$ eigenfunctions corresponding to the lowest eigenvalues $0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{\bar{M}}$ of $\mathbf{L}$, without taking into account the corresponding eigenfunction to the zero-eigenvalue, i.e.:

$$\mathbf{x} \to (f_1(\mathbf{x}), \ldots, f_{\bar{M}}(\mathbf{x})).$$

We have just argued that LE is not only a good method for embedding points in a graph in $\mathbb{R}^M$, but also works properly, without changing the algorithm proposed, if the sample data set is included in a smooth manifold of a lower dimension than the one of the original space.

2.2.3 Advantages and Disadvantages

LE is an algorithm with many advantages:

- LE lets us interpret our data in a geometric way. We have shown in Section 2.2.1 that we can reduce the dimension of the data while preserving its local geometry.
- The locality-preserving character of the LE algorithm makes it insensitive to outliers and noise.
- It exhibits stability with respect to the embedding, because this approach is based on the intrinsic geometric structure of the embedded manifold. This means that we will obtain the same embedding for the same underlying manifold even if it is included in spaces of very different dimension.
- Finally, it is simple to implement as it basically requires to solve an eigenvalue problem. The search of neighbor points could make the algorithm less efficient but there exist optimal methods as the one presented in Indyk [2000].

But it is not a perfect algorithm as it also presents some disadvantages:

- If we receive a new sample point, we should in principle repeat the whole algorithm over the new complete sample to reduce its dimension, because we would need to recalculate the eigenvalue problem.

- It is difficult to select values for the parameters $\bar{m}$, the reduced dimension, and t, the Heat Kernel parameter as they are data-dependent.
- Finally, the approximation presented only handles manifolds from which data is sampled uniformly, but this rarely happens in real applications.

2.3 Locally Linear Embedding

Locally Linear Embedding (LLE) is a method centered on preserving neighborhood relations [Roweis and Saul, 2000]. This point of view avoids the requirement of a distance computation between all the pairs of points that usually appears in other classical approaches. The main idea of this algorithm is the reconstruction of the nonlinear global structure from some local linear information known in advance.

To formalize this method, assume that we have a sample of n real vectors $\mathbf{x}^{(i)} \in \mathbb{R}^m$. We hope that the points $\mathbf{x}^{(i)}$ are located in or near a smooth low-dimensional manifold $\mathscr{M}$. The algorithm, briefly described below, builds a reconstruction method based on its local knowledge.

Step 1: Selecting the neighbors. First of all, we must obtain the local underlying information, and to do so we look for the κ nearest neighbors of each point $\mathbf{x}^{(i)}$ in the sample, that we will denote as $\mathbf{x_i}^{(j)}$. There exist different methods to reach this goal, like the κ-NN graph or the ϵ-neighborhood graph, that are based on the minimum distance to the main point and were briefly presented in Section 2.2.

Step 2: Reconstructing the graph with linear weights. To reconstruct each point with the information of its neighbors, an orthogonal projection of the points $\mathbf{x}^{(i)}$ is made over the affine linear span of the nearest data. This projection helps us to define the following cost function based on the reconstruction errors:

$$\mathcal{J}(\mathbf{W}) = \sum_i |\mathbf{x}^{(i)} - \sum_j w_{ij}\mathbf{x_i}^{(j)}|^2.$$

The weights w_{ij} symbolize the contribution of the neighbors $\mathbf{x_i}^{(j)}$ to the reconstruction of the point $\mathbf{x}^{(i)}$. Our aim is to find the values of w_{ij} that minimize the cost function. The minimization

problem can be defined as:

$$\arg\min_{\mathbf{W}} \left\{ \sum_i |\mathbf{x}^{(i)} - \sum_j w_{ij} \mathbf{x_i}^{(j)}|^2 \right\}$$
$$s.t. \begin{cases} w_{ij} = 0 \quad \text{if } \mathbf{x}^{(j)} \notin \{\mathbf{x_i}^{(k)}\} \\ \sum_i w_{ij} = 1 \quad \forall i \\ w_{ij} > 0. \end{cases}$$

Note that the third constraint is not mandatory.

Step 3: Mapping into the embedding coordinates. The last step of this algorithm consists in the construction of a neighborhood map for each point, such that the neighborhood in the space of dimension M is transformed into some global internal coordinates of the $\bar{M}$-dimensional embedded manifold. A good approximation for this map is a linear transformation: a rotation, rescaling or translation of the sample, and LLE builds it taking into account the information given by the reconstruction weights w_{ij}. The LLE map is defined as $\mathcal{f} : \mathbf{x}^{(i)} \rightarrow \mathbf{y}^{(i)}$ and has the following cost function:

$$\mathcal{J}(\mathbf{Y}) = \sum_i |\mathbf{y}^{(i)} - \sum_j w_{ij} \mathbf{y_i}^{(j)}|^2,$$

where the weights $\mathbf{W}$ are the ones obtained in Step 2.

Therefore, the best embedding $\mathbf{Y}$ is founded solving the problem of minimizing the cost function:

$$\arg\min_{\mathbf{Y}} \left\{ \sum_i |\mathbf{y}^{(i)} - \sum_j w_{ij} \mathbf{y_i}^{(j)}|^2 \right\}$$
$$s.t. \begin{cases} \sum_i \mathbf{y}^{(i)} = \mathbf{0} \\ \frac{1}{N} \sum_i \mathbf{y}^{(i)} \bigotimes \mathbf{y}^{(i)} = \mathbf{I}, \end{cases}$$

where $\mathbf{I}$ denotes the $N \times N$ identity matrix. Note that the first constrain implies that the coordinates are centered at the origin, and the second constrain avoids degenerate solutions, because we force the embedding points to have unit covariance.

In Roweis and Saul [2000] it is shown that the cost function can be rewritten in a simplified form as

$$\sum_i |\mathbf{y}^{(i)} - \sum_j w_{ij} \mathbf{y_i}^{(j)}|^2 = \sum_{i,j} m_{ij} (\mathbf{y}^{(i)} \mathbf{y_i}^{(j)}),$$

where we have introduced the matrix $\mathbf{M}$, that is defined as $\mathbf{M} = (\mathbf{I} - \mathbf{W})^\top (\mathbf{I} - \mathbf{W})$, with components given by:

$$m_{ij} = \delta_{ij} - w_{ij} - w_{ji} + \sum_l w_{li} w_{lj}.$$

Thus, to minimize the cost function $\mathcal{J}(\mathbf{Y})$ is equivalent to minimize $\mathbf{Y}^\top \mathbf{M}\mathbf{Y}$, and this is equivalent to find the eigenvectors of the sparse matrix $\mathbf{M}$, which has the property of being a symmetric positive semidefinite matrix. The optimal embedding is determined by the $\bar{m}$ eigenvectors corresponding to the minor eigenvalues of this matrix, after discarding the zero-eigenvalue. As we have done in LE, we discard this value because it corresponds to a trivial solution.

2.3.1 LLE Laplacian Point of View

The LLE algorithm explained above can be also studied in the same Laplacian terms as the LE algorithm in Section 2.2 [Belkin and Nyogi, 2003]. This change of point of view only affects to the last step: the embedding of the original coordinates, which are determined by the matrix

$$\mathbf{M} = (\mathbf{I} - \mathbf{W})^\top (\mathbf{I} - \mathbf{W}),$$

which can be approximately rewritten as

$$\mathbf{M}f \approx \frac{1}{2}\mathbf{L}^2 f.$$

The proof of this approximation is formally presented in Belkin and Nyogi [2003], and can be summarized in the following three steps:

1. In a first step it can be proved that, for a fixed point $\mathbf{x}^{(i)}$,

$$[(\mathbf{I} - \mathbf{W})f]_i \approx -\frac{1}{2}\sum_j m_{ij}(\mathbf{x}^{(i)} - \mathbf{x_i}^{(j)})^\top \mathbf{H}^{(i)}(\mathbf{x}^{(i)} - \mathbf{x_i}^{(j)}),$$

   where $\mathbf{H}$ represents the Hessian matrix of f.

2. Defining $\mathbf{v}^{(j)} = \mathbf{x_i}^{(j)} - \mathbf{x}^{(i)}$ and assuming that $\sqrt{w_{ij}}\mathbf{v}_i$ form an orthonormal basis, it can be proven that

$$\mathrm{E}(\mathbf{v}^\top \mathbf{H}\mathbf{v}) = r\mathbf{L}f,$$

   where $r = \mathrm{E}(< \mathbf{v}^{(i)}, e_i >^2)$, which is independent of i, being e_i an orthonormal basis for the Hessian matrix $\mathbf{H}$.

3. Putting together Steps 1 and 2, we arrive to

$$(\mathbf{I} - \mathbf{W})^\top (\mathbf{I} - \mathbf{W})f \approx \frac{1}{2}\mathbf{L}^2 f.$$

As shown at the beginning of this section, LLE tries to minimize the function $f^\top (\mathbf{I} - \mathbf{W})^\top (\mathbf{I} - \mathbf{W})f$, and we have seen that minimizing this function is equivalent to look for the eigendecomposition of the matrix $(\mathbf{I} - \mathbf{W})^\top (\mathbf{I} - \mathbf{W})$. We have just discussed how this expression is

equivalent to the matrix $\frac{1}{2}\mathbf{L}^2$. Consequently, to obtain an embedding of the original points following the LLE algorithm we can just look for the eigenvectors of $\mathbf{L}^2$, that coincide with those of $\mathbf{L}$.

2.4 Spectral Clustering

Clustering is one of the most widely used techniques for data analysis. In recent years, Spectral Clustering (SC) has become one of the most popular modern clustering algorithms [Luxburg, 2007]. These methods are easy to implement and can be solved efficiently, although at the beginning it could seem to be a little bit hard to understand how and why they work. In the end they are able to extract the geometry and local information from the data set we are working with in order to, first, reduce dimension and, later, apply any of the existing clustering algorithms that have a good performance in easier subspaces. We have seen this idea in the previous sections when we have explained the LE algorithm.

2.4.1 Different SC Algorithms

There exist some different algorithms to implement SC, and the main difference between them is the graph Laplacian they employ to implement the embedding. As we will see, all these algorithms present the same structure: they receive as input an arbitrary subset $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$, with $\mathbf{x}^{(i)} \in \mathbb{R}^M$, all the algorithms use the weight matrix $\mathbf{W}$ with $w_{ij} = \mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ according to some symmetric non-negative kernel similarity function, and the clusters in the original space are always the result of the algorithm. The main trick in all of them is the change of the data $\{\mathbf{x}^{(i)}\}_{i=1}^N$ from the original space to the points $\{\mathbf{y}^{(i)}\}_{i=1}^N$ with $\mathbf{y}^{(i)} \in \mathbb{R}^{\bar{M}}$ lying in a Euclidean space of smaller dimension. This representation change is very useful, because it is easier to work in Euclidean spaces, and the embedding does not change the structural properties of the initial space.

2.4.1.1 Unnormalized Spectral Clustering

Algorithm 2.1 presents the easiest SC algorithm, which is based on the unnormalized graph Laplacian to reduce the dimension of the original space. Recall that the unnormalized Laplacian operator was defined in Definition 2.2 as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}.$$

ALGORITHM 2.1: Unnormalized Spectral Clustering Algorithm.

Input: $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}$; κ, the number of clusters ;

Output: Clusters $\mathscr{A}_1, \ldots, \mathscr{A}_k$ where $\mathscr{A}_i = \{j | \mathbf{y}^{(j)} \in \mathscr{C}_i\}$;

1: Construct a similarity graph $\mathscr{G}$, with adjacency matrix $\mathbf{W}$;
2: $\mathbf{L} = \mathbf{D} - \mathbf{W}$; ▶ *Unnormalized Graph Laplacian.*
3: Obtain $\vartheta_1, \ldots, \vartheta_{\bar{M}}$; ▶ $\bar{M}$ *First Eigenvectors of* $\mathbf{L}$.
4: **for** $i = 1$ to N **do**
5: Compute $\mathbf{y}^{(i)} \in \mathbb{R}^{\bar{M}}$ as the i-th row of ϑ ; ▶ *Embedding.*
6: **end for**
7: $\mathscr{C}_1, \ldots, \mathscr{C}_{\bar{M}} = \kappa$-means$(\mathbf{y}^{(i)})$;

2.4.1.2 Normalized Spectral Clustering

In this subsection we present two different versions of the SC algorithm, according to the two different types of normalized graph Laplacian that exist and that we define below.

Definition 2.3 (Symmetric Graph Laplacian)**.** The *Symmetric Graph Laplacian* is given by the expression

$$\begin{aligned}\mathbf{L}_{sym} &= \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} \\ &= \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}.\end{aligned}$$

Definition 2.4 (Random Walk Graph Laplacian)**.** The *Random Walk Graph Laplacian* is closely related with random walks, and it is defined as

$$\begin{aligned}\mathbf{L}_{rw} &= \mathbf{D}^{-1}\mathbf{L} \\ &= \mathbf{I} - \mathbf{D}^{-1}\mathbf{W},\end{aligned}$$

where $\mathbf{D}^{-1}\mathbf{W}$ is now a Markov matrix.

The algorithm associated to the normalized symmetric graph Laplacian (Definition 2.3), is presented in Algorithm 2.2. As we will explained in Section 2.4.2.1, this algorithm needs an extra normalization step (step 4 of the algorithm) that the other algorithms do not require.

The algorithm associated to the normalized random walk graph Laplacian (see Definition 2.4) is presented in Algorithm 2.2. Notice that this third algorithm is essentially the LE algorithm

ALGORITHM 2.2: Normalized SC Algorithm: Symmetric Graph Laplacian.

Input: $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}$; κ, the number of clusters ;

Output: Clusters $\mathscr{A}_1, \ldots, \mathscr{A}_k$ where $\mathscr{A}_i = \{j | \mathbf{y}^{(j)} \in \mathscr{C}_i\}$;

1: Construct a similarity graph $\mathscr{G}$, with adjacency matrix $\mathbf{W}$;

2: $\mathbf{L}_{sym} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$; ▶ *Normalized Symmetric Graph* $\mathbf{L}$

3: Obtain the eigenvector $\vartheta_1, \ldots, \vartheta_{\bar{M}}$; ▶ $\bar{M}$ *First Eigenvectors of* $\mathbf{L}_{sym}$.

4: Compute $u_{ij} = \frac{\vartheta_{ij}}{\left(\sum_\ell \vartheta_{i\ell}^2\right)^{\frac{1}{2}}}$; ▶ *Normalized matrix.*

5: **for** $i = 1$ to N **do**

6: Compute $\mathbf{y}^{(i)} \in \mathbb{R}^{\bar{M}}$ as the i-th row of $\mathbf{U}$; ▶ *Embedding.*

7: **end for**

8: $\mathscr{C}_1, \ldots, \mathscr{C}_{\bar{M}} = \kappa$-means$(\mathbf{y}^{(i)})$;

ALGORITHM 2.3: Normalized SC Algorithm: Random Walk Graph Laplacian.

Input: $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}$; κ, the number of clusters ;

Output: Clusters $\mathscr{A}_1, \ldots, \mathscr{A}_k$ where $\mathscr{A}_i = \{j | \mathbf{y}^{(j)} \in \mathscr{C}_i\}$;

1: Construct a similarity graph $\mathscr{G}$, with adjacency matrix $\mathbf{W}$;

2: $\mathbf{L} = \mathbf{D} - \mathbf{W}$; ▶ *Unnormalized Graph Laplacian.*

3: Solve the problem $\mathbf{L}\vartheta = \lambda\mathbf{D}\vartheta$ and obtain the first $\bar{M}$ eigenvectors $\vartheta_1, \ldots, \vartheta_{\bar{M}}$;

4: **for** $i = 1$ to N **do**

5: Compute $\mathbf{y}^{(i)} \in \mathbb{R}^{\bar{M}}$ as the i-th row of ϑ ; ▶ *Embedding.*

6: **end for**

7: $\mathscr{C}_1, \ldots, \mathscr{C}_{\bar{M}} = \kappa$-means$(\mathbf{y}^{(i)})$;

plus κ-means, and in its pseudocode it is assumed that the eigenvalues obtained as solution of $\mathbf{L}\vartheta = \lambda\mathbf{D}\vartheta$ are equivalent to the eigenvalues of $\mathbf{L}_{rw}$. We can easily prove this statement.

Lemma 2.2 ($\mathbf{L}_{rw}$ Eigendecomposition)**.** *λ is an eigenvalue of $\mathbf{L}_{rw}$ with ϑ its corresponding eigenvector $\Leftrightarrow$ λ and ϑ solves the equation $\mathbf{L}\vartheta = \lambda\mathbf{D}\vartheta$.*

Proof. Assume that λ is an eigenvalue of $\mathbf{L}_{rw}$ and ϑ its corresponding eigenvector. Then we have

$$\begin{aligned}\lambda\vartheta &= \mathbf{L}_{rw}\vartheta \\ &= \mathbf{D}^{-1}\mathbf{L}\vartheta\end{aligned}$$

and, therefore, we can conclude that

$$\mathbf{L}\vartheta = \lambda \mathbf{D}\vartheta. \qquad \square$$

2.4.2 Justification

The main idea of clustering is to group data in sets with similar properties, which is also the purpose of the algorithms previously presented. In this new subsection we will try to justify from different points of view that these SC algorithms cluster properly the original data based on the information they extract when building the embedding space.

2.4.2.1 Graph Cut Point of View

The graph cut theory can give us a first idea of why SC is a good data clustering technique. The fundamental graph cut theory consists in the construction of a graph partition such that the edges between groups present low weights, but inside each group, the connections between nodes are strong. To apply this concept to construct partitions over a graph we need to define a new measure: the cut of the graph.

Definition 2.5 (Graph Cut)**.** Let $\mathscr{A}, \mathscr{B} \subset \mathscr{S}$, where $\mathscr{A} \bigcap \mathscr{B} = \emptyset$. Then, we define the *cut* of the graph across $\mathscr{A}$ and $\mathscr{B}$ as

$$\text{cut}(\mathscr{A}, \mathscr{B}) = \sum_{\substack{i \in \mathscr{A} \\ j \in \mathscr{B}}} w_{ij}.$$

We are interested in obtaining disjoint subsets, and this occurs when $\text{cut}(\mathscr{A}, \mathscr{B}) = 0$. The cut of the graph gives us an estimation about which edges should be eliminated from $\mathscr{A}$ to isolate it from $\mathscr{B}$.

If $\mathscr{G}$ is the similarity graph and $\mathbf{W}$ the adjacency matrix, to find a partition of the graph is the same as solving the problem of finding a partition with a minimum cut. More generally, we have to find the partition $\{\mathscr{A}_1, \ldots, \mathscr{A}_\kappa\}$ that minimizes

$$\text{cut}(\mathscr{A}_1, \ldots, \mathscr{A}_\kappa) = \sum_{i=1}^{\kappa} \text{cut}(\mathscr{A}_i, \bar{\mathscr{A}_i}),$$

where $\bar{\mathscr{A}_i}$ represents the complement of $\mathscr{A}_i$.

In practice, minimizing this quantity does not give a good partition of the data. To sort out this problem we should be more strict, like for example forcing the partition sets $\{\mathscr{A}_1, \ldots, \mathscr{A}_\kappa\}$ to be reasonably large. We define then two new, stricter objective functions whose minimization gives rise to a good partition. These new measures are the RatioCut [Hagen and Kahng, 1992] and the NCut [Shi and Malik, 2000].

Definition 2.6 (RatioCut)**.** Let $\mathscr{A}_i \subset \mathscr{S}$ $i = 1 \dots \kappa$, with $|\mathscr{A}_i|$ denoting the number of nodes in $\mathscr{A}_i$. Then, we define the *RatioCut* of the graph partition as

$$\text{RatioCut}(\mathscr{A}_1, \dots, \mathscr{A}_\kappa) = \sum_{i=1}^{\kappa} \frac{\text{cut}(\mathscr{A}_i, \bar{\mathscr{A}_i})}{|\mathscr{A}_i|}. \tag{2.11}$$

Definition 2.7 (NCut)**.** Let $\mathscr{A}_i \subset \mathscr{S}$ $i = 1 \dots \kappa$, with $\text{vol}(\mathscr{A}_i) = \sum_{i \in \mathscr{A}_i} \sum_{j=1}^{N} w_{ij}$ representing the weights of $\mathscr{A}_i$ edges. Then, we define the *Normalized Cut (NCut)* of the graph partition as

$$\text{NCut}(\mathscr{A}_1, \dots, \mathscr{A}_\kappa) = \sum_{i=1}^{\kappa} \frac{\text{cut}(\mathscr{A}_i, \bar{\mathscr{A}_i})}{\text{vol}(\mathscr{A}_i)}. \tag{2.12}$$

The RatioCut (Equation (2.11)) is related with the unnormalized SC, and it measures the partition size using the number of vertices on the graph. The NCut (Equation (2.12)) is related with the normalized SC, and it measures the partition size via the weights of its edges. In both cases, if $\mathscr{A}_i$ is very small, the objective function will have a high value. As an approximation of the difference between these two measures, the RatioCut and the NCut, we can consider their related problems:

$$\min \left\{ \sum_{i=1}^{\kappa} \frac{1}{|\mathscr{A}_i|} \right\} \tag{2.13}$$

$$\min \left\{ \sum_{i=1}^{\kappa} \frac{1}{\text{vol}(\mathscr{A}_i)} \right\}, \tag{2.14}$$

that achieve the minimum when all clusters have the same $|\mathscr{A}_i|$ in the case of Problem (2.13) or the same $\text{vol}(\mathscr{A}_i)$ for Problem (2.14). Thus we may interpret that in the RatioCut and NCut problems we are searching for balanced clusters. In any case both are NP-Hard problems [Wagner and Wagner, 1993] and SC will solve relaxed versions of them. We next describe it in both cases.

Approximation to RatioCut. First, we are going to study the approximation to the RatioCut with $\kappa = 2$. The main objective is to solve the minimization problem given by

$$\min_{\mathscr{A} \subset \mathscr{S}} \{\text{RatioCut}(\mathscr{A}, \bar{\mathscr{A}})\}.$$

For this purpose, we rewrite the problem assuming $\mathscr{A} \subset \mathscr{S}$ and considering

$$f = (f_1, \dots, f_N)^\top \in \mathbb{R}^N \text{ with entries } f_i = \begin{cases} \sqrt{\frac{|\bar{\mathscr{A}}|}{|\mathscr{A}|}} & \mathbf{x}^{(i)} \in \mathscr{A} \\ -\sqrt{\frac{|\mathscr{A}|}{|\bar{\mathscr{A}}|}} & \mathbf{x}^{(i)} \in \bar{\mathscr{A}}. \end{cases}$$

By rewriting the objective function using the normalized graph Laplacian, it is easy to prove that minimizing $f^\top \mathbf{L} f$ is equivalent to minimizing the original RatioCut problem. To do so, recall what we have proved in Lemma 2.1: $f^\top \mathbf{L} f = \sum_{i,j=1}^{N} w_{ij}(f_i - f_j)^2$. Therefore,

$$\begin{aligned}
f^\top \mathbf{L} f &= \sum_{i,j=1}^{N} w_{ij}(f_i - f_j)^2 \\
&= \sum_{\substack{i\in\mathscr{A}\\ j\in\bar{\mathscr{A}}}} w_{ij}\left(\sqrt{\frac{|\bar{\mathscr{A}}|}{|\mathscr{A}|}} + \sqrt{\frac{|\mathscr{A}|}{|\bar{\mathscr{A}}|}}\right)^2 + \sum_{\substack{i\in\bar{\mathscr{A}}\\ j\in\mathscr{A}}} w_{ij}\left(-\sqrt{\frac{|\bar{\mathscr{A}}|}{|\mathscr{A}|}} - \sqrt{\frac{|\mathscr{A}|}{|\bar{\mathscr{A}}|}}\right)^2 \\
&+ \sum_{\substack{i\in\mathscr{A}\\ j\in\mathscr{A}}} w_{ij}\left(\sqrt{\frac{|\bar{\mathscr{A}}|}{|\mathscr{A}|}} - \sqrt{\frac{|\bar{\mathscr{A}}|}{|\mathscr{A}|}}\right)^2 + \sum_{\substack{i\in\bar{\mathscr{A}}\\ j\in\bar{\mathscr{A}}}} w_{ij}\left(-\sqrt{\frac{|\bar{\mathscr{A}}|}{|\mathscr{A}|}} + \sqrt{\frac{|\bar{\mathscr{A}}|}{|\mathscr{A}|}}\right)^2 \\
&= \sum_{\substack{i\in\mathscr{A}\\ j\in\bar{\mathscr{A}}}} w_{ij}\left(\sqrt{\frac{|\bar{\mathscr{A}}|}{|\mathscr{A}|}} + \sqrt{\frac{|\mathscr{A}|}{|\bar{\mathscr{A}}|}}\right)^2 + \sum_{\substack{i\in\bar{\mathscr{A}}\\ j\in\mathscr{A}}} w_{ij}\left(-\sqrt{\frac{|\bar{\mathscr{A}}|}{|\mathscr{A}|}} - \sqrt{\frac{|\mathscr{A}|}{|\bar{\mathscr{A}}|}}\right)^2 \\
&= 2\text{cut}(\mathscr{A}, \bar{\mathscr{A}})\left(\frac{|\bar{\mathscr{A}}|}{|\mathscr{A}|} + \frac{|\mathscr{A}|}{|\bar{\mathscr{A}}|} + 2\right) \\
&= 2\text{cut}(\mathscr{A}, \bar{\mathscr{A}})\left(\frac{|\bar{\mathscr{A}}| + |\mathscr{A}|}{|\mathscr{A}|} + \frac{|\mathscr{A}| + |\bar{\mathscr{A}}|}{|\bar{\mathscr{A}}|}\right) \\
&= 2(|\bar{\mathscr{A}}| + |\mathscr{A}|)\left(\frac{\text{cut}(\mathscr{A}, \bar{\mathscr{A}})}{|\mathscr{A}|} + \frac{\text{cut}(\mathscr{A}, \bar{\mathscr{A}})}{|\bar{\mathscr{A}}|}\right) \\
&= 2|\mathscr{S}|\text{RatioCut}(\mathscr{A}, \bar{\mathscr{A}}).
\end{aligned}$$

We can now rewrite the minimization problem in the form

$$\begin{aligned}
&\min_{\mathscr{A}\subset\mathscr{S}} \{f^\top \mathbf{L} f\} \\
&s.t.\ f_i = \begin{cases} \sqrt{\frac{|\bar{\mathscr{A}}|}{|\mathscr{A}|}} & \mathbf{x}^{(i)} \in \mathscr{A} \\ -\sqrt{\frac{|\mathscr{A}|}{|\bar{\mathscr{A}}|}} & \mathbf{x}^{(i)} \in \bar{\mathscr{A}}. \end{cases}
\end{aligned} \tag{2.15}$$

With this trick we have discretized the RatioCut equation, but the problem continues being NP-Hard as we are minimizing over $\mathscr{A}$. In order to solve this, we can study some characteristic of the function f that we have just defined. We should first notice that $f^\top \mathbf{1} = 0$:

$$\begin{aligned}
f^\top \mathbf{1} = \sum_{i=1}^{N} f_i &= \sum_{i\in\mathscr{A}} \sqrt{\frac{|\bar{\mathscr{A}}|}{|\mathscr{A}|}} - \sum_{i\in\bar{\mathscr{A}}} \sqrt{\frac{|\mathscr{A}|}{|\bar{\mathscr{A}}|}} \\
&= |\mathscr{A}|\sqrt{\frac{|\bar{\mathscr{A}}|}{|\mathscr{A}|}} - |\bar{\mathscr{A}}|\sqrt{\frac{|\mathscr{A}|}{|\bar{\mathscr{A}}|}} = \sqrt{|\mathscr{A}||\bar{\mathscr{A}}|} - \sqrt{|\bar{\mathscr{A}}||\mathscr{A}|} = 0.
\end{aligned}$$

We observe next that $\|\boldsymbol{f}\|^2 = |\mathscr{S}|$, because

$$\begin{aligned}\|\boldsymbol{f}\|^2 = \sum_{i=1}^{N} f_i^2 &= |\mathscr{A}|\frac{|\bar{\mathscr{A}}|}{|\mathscr{A}|} + |\bar{\mathscr{A}}|\frac{|\mathscr{A}|}{|\bar{\mathscr{A}}|} \\ &= |\bar{\mathscr{A}}| + |\mathscr{A}| = N = |\mathscr{S}|.\end{aligned}$$

Considering these two conditions, Problem (2.15) is now equivalent to

$$\min_{\mathscr{A}\subset\mathscr{S}} \{\boldsymbol{f}^\top \mathbf{L}\boldsymbol{f}\}$$
$$s.t. \begin{cases} \boldsymbol{f} \perp \mathbf{1} \\ f_i = \begin{cases} \sqrt{\frac{|\bar{\mathscr{A}}|}{|\mathscr{A}|}} & \mathbf{x}^{(i)} \in \mathscr{A} \\ -\sqrt{\frac{|\mathscr{A}|}{|\bar{\mathscr{A}}|}} & \mathbf{x}^{(i)} \in \bar{\mathscr{A}} \end{cases} \\ \|\boldsymbol{f}\| = \sqrt{N}. \end{cases}$$

To simplify the problem definition, we can apply a constrain relaxation, consisting on the elimination of the restriction over the discrete values of f_i, allowing $f_i \in \mathbb{R}$.

$$\min_{\boldsymbol{f}\in\mathbb{R}^N} \{\boldsymbol{f}^\top \mathbf{L}\boldsymbol{f}\} \tag{2.16}$$
$$s.t. \begin{cases} \boldsymbol{f} \perp \mathbf{1} \\ \|\boldsymbol{f}\| = \sqrt{N}. \end{cases}$$

Applying the Rayleigh–Ritz theorem, the solution is given by the vector $\boldsymbol{f}^{\text{op}}$ which coincides with the eigenvector that corresponds to the lowest non-zero eigenvalue λ_1 of $\mathbf{L}$. To obtain a partition of the graph, we transform $\boldsymbol{f}$ in a discrete indicator

$$\boldsymbol{f} = \begin{cases} \mathbf{x}^{(i)} \in \mathscr{A} & f_i \geqslant 0 \\ \mathbf{x}^{(i)} \in \bar{\mathscr{A}} & f_i < 0. \end{cases}$$

We have seen in Section 2.2 that solving Problem (2.16) is equivalent to find the eigenvector of the generalized eigenvalue problem $\mathbf{L}\boldsymbol{f} = \lambda\mathbf{D}\boldsymbol{f}$. Note that we can obtain the same solution of the relaxed RatioCut problem by applying the unnormalized SC algorithm shown in Algorithm 2.1.

We can extend the previous explanation to an arbitrary κ in the following way. Let $\{\mathscr{A}_1, \ldots, \mathscr{A}_\kappa\}$ be a partition of $\mathscr{S}$. We define κ indicator vectors $\mathbf{h}_i = (h_{1i}, \ldots, h_{Ni})^\top$ such that

$$h_{ji} = \begin{cases} \frac{1}{\sqrt{|\mathscr{A}i|}} & j \in \mathscr{A}i \\ 0 & \text{otherwise.} \end{cases}$$

$\mathbf{H} \in \mathbb{R}^{N \times \kappa}$ will be the matrix that contains these indicator vectors $\mathbf{h}_i$ on their columns. It is easy to see that this matrix has the property of being an orthonormal matrix ($\mathbf{H}^\top \mathbf{H} = \mathbf{I}$).

In this case, remembering that $\mathbf{W}$ is a symmetric matrix, the objective function can be rewritten in the following way

$$\begin{aligned}
\mathbf{h}_i^\top \mathbf{L}\mathbf{h}_i &= \sum_{\ell,m=1}^{N} w_{\ell m}(h_{\ell i} - h_{mi})^2 \\
&= \sum_{\ell,m \in \mathscr{A}i} w_{\ell m}(h_{\ell i} - h_{mi})^2 + \sum_{\substack{\ell \in \mathscr{A}i \\ m \in \bar{\mathscr{A}_i}}} w_{\ell m}(h_{\ell i} - h_{mi})^2 \\
&\quad + \sum_{\substack{\ell \in \bar{\mathscr{A}_i} \\ m \in \mathscr{A}i}} w_{\ell m}(h_{\ell i} - h_{mi})^2 + \sum_{\ell,m \in \bar{\mathscr{A}_i}} w_{\ell m}(h_{\ell i} - h_{mi})^2 \\
&= \sum_{\ell,m \in \mathscr{A}i} w_{\ell m} \left(\frac{1}{\sqrt{|\mathscr{A}_i|}} - \frac{1}{\sqrt{|\mathscr{A}_i|}} \right)^2 + \sum_{\substack{\ell \in \mathscr{A}_i \\ m \in \bar{\mathscr{A}_i}}} w_{\ell m} \left(\frac{1}{\sqrt{|\mathscr{A}_i|}} - 0 \right)^2 \\
&\quad + \sum_{\substack{\ell \in \bar{\mathscr{A}_i} \\ m \in \mathscr{A}_i}} w_{\ell m} \left(0 - \frac{1}{\sqrt{|\mathscr{A}_i|}} \right)^2 + \sum_{\ell,m \in \bar{\mathscr{A}_i}} w_{\ell m} \cdot 0 \\
&= 2 \sum_{\substack{\ell \in \mathscr{A}_i \\ m \in \bar{\mathscr{A}_i}}} w_{\ell m} \left(\frac{1}{\sqrt{|\mathscr{A}_i|}} \right)^2 \\
&= 2\text{cut}(\mathscr{A}_i, \bar{\mathscr{A}_i}) \frac{1}{|\mathscr{A}_i|},
\end{aligned}$$

which is equivalent in matrix notation to

$$\mathbf{h}_i^\top \mathbf{L}\mathbf{h}_i = (\mathbf{H}^\top \mathbf{L}\mathbf{H})_{ii}.$$

Consequently, the RatioCut can be rewritten as

$$\begin{aligned}
\text{RatioCut}(\mathscr{A}_1, \ldots, \mathscr{A}_\kappa) &= \frac{1}{2} \sum_{i=1}^{\kappa} \mathbf{h}_i^\top \mathbf{L}\mathbf{h}_i \\
&= \frac{1}{2} \sum_{i=1}^{\kappa} (\mathbf{H}^\top \mathbf{L}\mathbf{H})_{ii} \\
&= \frac{1}{2} \text{Tr}(\mathbf{H}^\top \mathbf{L}\mathbf{H}),
\end{aligned}$$

and then the minimization problem to be solved is

$$\min_{\mathscr{A}_1,\cdots,\mathscr{A}_\kappa}\{\mathrm{Tr}(\mathbf{H}^\top\mathbf{L}\mathbf{H})\}$$
$$s.t.\begin{cases}\mathbf{H}^\top\mathbf{H}=\mathbf{I}\\ h_{ji}=\begin{cases}\frac{1}{\sqrt{|\mathscr{A}_i|}}, & j\in\mathscr{A}_i\\ 0 & \text{otherwise.}\end{cases}\end{cases}$$

In this case, we can relax the problem allowing any real entry in matrix $\mathbf{H}$, arriving at

$$\min_{\mathbf{H}\in\mathbb{R}^{N\times\kappa}}\{\mathrm{Tr}(\mathbf{H}^\top\mathbf{L}\mathbf{H})\}$$
$$s.t.\ \mathbf{H}^\top\mathbf{H}=\mathbf{I}.$$

Thanks again to the Rayleigh–Ritz theorem we know that a solution will be a matrix $\mathbf{H}^{\mathrm{op}}$ whose columns will coincide with the first κ eigenvectors of $\mathbf{L}$ corresponding to non-zero eigenvalues. Again we should discretized this result, but now we cannot apply the previous procedure; as an alternative we can apply κ-means over the matrix rows [Luxburg, 2007]. Thus, the result obtain solving the general relaxed RatioCut problem is equivalent to the one obtained when applying the unnormalized SC algorithm (Algorithm 2.1).

Approximation to NCut. The approach to the NCut problem is very similar to the one just explained for the RatioCut. We are going to start again with the easiest case: $\kappa=2$. First of all, we should define a cluster indicator vector

$$f_i=\begin{cases}\sqrt{\frac{\mathrm{vol}(\bar{\mathscr{A}})}{\mathrm{vol}(\mathscr{A})}} & \mathbf{x}^{(i)}\in\mathscr{A}\\ -\sqrt{\frac{\mathrm{vol}(\mathscr{A})}{\mathrm{vol}(\bar{\mathscr{A}})}} & \mathbf{x}^{(i)}\in\bar{\mathscr{A}}.\end{cases}$$

This indicator is closely related to the NCut defined before in Equation (2.12), as we have

$$
\begin{aligned}
f^\top \mathbf{L} f &= \sum_{i,j=1}^{N} w_{ij}(f_i - f_j)^2 \\
&= \sum_{\substack{i\in\mathscr{A}\\ j\in\bar{\mathscr{A}}}} w_{ij}\left(\sqrt{\frac{\mathrm{vol}(\bar{\mathscr{A}})}{\mathrm{vol}(\mathscr{A})}} + \sqrt{\frac{\mathrm{vol}(\mathscr{A})}{\mathrm{vol}(\bar{\mathscr{A}})}}\right)^2 + \sum_{\substack{i\in\bar{\mathscr{A}}\\ j\in\mathscr{A}}} w_{ij}\left(-\sqrt{\frac{\mathrm{vol}(\mathscr{A})}{\mathrm{vol}(\bar{\mathscr{A}})}} - \sqrt{\frac{\mathrm{vol}(\bar{\mathscr{A}})}{\mathrm{vol}(\mathscr{A})}}\right)^2 \\
&+ \sum_{i,j\in\mathscr{A}} w_{ij}\left(\sqrt{\frac{\mathrm{vol}(\bar{\mathscr{A}})}{vol(\mathscr{A})}} - \sqrt{\frac{\mathrm{vol}(\bar{\mathscr{A}})}{\mathrm{vol}(\mathscr{A})}}\right)^2 + \sum_{i,j\in\bar{\mathscr{A}}} w_{ij}\left(-\sqrt{\frac{\mathrm{vol}(\mathscr{A})}{\mathrm{vol}(\bar{\mathscr{A}})}} + \sqrt{\frac{\mathrm{vol}(\mathscr{A})}{\mathrm{vol}(\bar{\mathscr{A}})}}\right)^2 \\
&= \sum_{\substack{i\in\mathscr{A}\\ j\in\bar{\mathscr{A}}}} w_{ij}\left(\sqrt{\frac{\mathrm{vol}(\bar{\mathscr{A}})}{\mathrm{vol}(\mathscr{A})}} + \sqrt{\frac{\mathrm{vol}(\mathscr{A})}{\mathrm{vol}(\bar{\mathscr{A}})}}\right)^2 + \sum_{\substack{i\in\bar{\mathscr{A}}\\ j\in\mathscr{A}}} w_{ij}\left(-\sqrt{\frac{\mathrm{vol}(\mathscr{A})}{\mathrm{vol}(\bar{\mathscr{A}})}} - \sqrt{\frac{\mathrm{vol}(\bar{\mathscr{A}})}{\mathrm{vol}(\mathscr{A})}}\right)^2 \\
&= 2\mathrm{cut}(\mathscr{A},\bar{\mathscr{A}})\left(\frac{\mathrm{vol}(\bar{\mathscr{A}})}{\mathrm{vol}(\mathscr{A})} + \frac{\mathrm{vol}(\mathscr{A})}{\mathrm{vol}(\bar{\mathscr{A}})} + 2\right) \\
&= 2\mathrm{cut}(\mathscr{A},\bar{\mathscr{A}})\left(\frac{\mathrm{vol}(\bar{\mathscr{A}}) + \mathrm{vol}(\mathscr{A})}{\mathrm{vol}(\mathscr{A})} + \frac{\mathrm{vol}(\mathscr{A}) + \mathrm{vol}(\bar{\mathscr{A}})}{\mathrm{vol}(\bar{\mathscr{A}})}\right) \\
&= 2\mathrm{vol}(\mathscr{S})\mathrm{NCut}(\mathscr{A},\bar{\mathscr{A}}).
\end{aligned}
$$

As we have previously done in the RatioCut example, we are going to look for some restrictions to the minimization problem that take into account relevant properties of f. First, $(\mathbf{D}f)^\top \mathbf{1} = 0$ as shown below:

$$
\begin{aligned}
(\mathbf{D}f)^\top \mathbf{1} &= \sum_i d_i f_i \\
&= \sum_{i\in\mathscr{A}} d_i f_i + \sum_{i\in\bar{\mathscr{A}}} d_i f_i \\
&= \sqrt{\frac{\mathrm{vol}(\bar{\mathscr{A}})}{\mathrm{vol}(\mathscr{A})}} \sum_{i\in\mathscr{A}} d_i - \sqrt{\frac{\mathrm{vol}(\mathscr{A})}{\mathrm{vol}(\bar{\mathscr{A}})}} \sum_{i\in\bar{\mathscr{A}}} d_i \\
&= \sqrt{\frac{\mathrm{vol}(\bar{\mathscr{A}})}{\mathrm{vol}(\mathscr{A})}} \sum_{i\in\mathscr{A}} \sum_{j=1}^{N} w_{ij} - \sqrt{\frac{\mathrm{vol}(\mathscr{A})}{\mathrm{vol}(\bar{\mathscr{A}})}} \sum_{i\in\bar{\mathscr{A}}} \sum_{j=1}^{N} w_{ij} \\
&= \sqrt{\frac{\mathrm{vol}(\bar{\mathscr{A}})}{\mathrm{vol}(\mathscr{A})}} \mathrm{vol}(\mathscr{A}) - \sqrt{\frac{\mathrm{vol}(\mathscr{A})}{\mathrm{vol}(\bar{\mathscr{A}})}} \mathrm{vol}(\bar{\mathscr{A}}) \\
&= \sqrt{\mathrm{vol}(\bar{\mathscr{A}})\mathrm{vol}(\mathscr{A})} - \sqrt{\mathrm{vol}(\mathscr{A})\mathrm{vol}(\bar{\mathscr{A}})} \\
&= 0.
\end{aligned}
$$

Another interesting property of f is that $f^\top \mathbf{D} f = \text{vol}(\mathscr{S})$, which can be derived following the next reasoning:

$$\begin{aligned} f^\top \mathbf{D} f &= \sum_{i \in \mathscr{A}} d_i f_i^2 + \sum_{i \in \bar{\mathscr{A}}} d_i f_i^2 \\ &= \sum_{i \in \mathscr{A}} d_i \frac{\text{vol}(\bar{\mathscr{A}})}{\text{vol}(\mathscr{A})} + \sum_{i \in \bar{\mathscr{A}}} d_i \frac{\text{vol}(\mathscr{A})}{\text{vol}(\bar{\mathscr{A}})} \\ &= \frac{\text{vol}(\bar{\mathscr{A}})}{\text{vol}(\mathscr{A})} \sum_{i \in \mathscr{A}} \sum_{j=1}^{N} w_{ij} + \frac{\text{vol}(\mathscr{A})}{\text{vol}(\bar{\mathscr{A}})} \sum_{i \in \bar{\mathscr{A}}} \sum_{j=1}^{N} w_{ij} \\ &= \text{vol}(\bar{\mathscr{A}}) + \text{vol}(\mathscr{A}) \\ &= \text{vol}(\mathscr{S}). \end{aligned}$$

Taking these expressions into consideration, we can now rewrite the minimization NCut problem (Equation (2.12)) as

$$\min_{\mathscr{A} \subset \mathscr{S}} \{f^\top \mathbf{L} f\} \quad s.t. \begin{cases} \mathbf{D} f \perp \mathbf{1} \\ f_i = \begin{cases} \sqrt{\frac{\text{vol}(\bar{\mathscr{A}})}{\text{vol}(\mathscr{A})}} & \mathbf{x}^{(i)} \in \mathscr{A} \\ -\sqrt{\frac{\text{vol}(\mathscr{A})}{\text{vol}(\bar{\mathscr{A}})}} & \mathbf{x}^{(i)} \in \bar{\mathscr{A}} \end{cases} \\ f^\top \mathbf{D} f = \text{vol}(\mathscr{S}). \end{cases}$$

And now, as done in previous approximations, we can relax the problem allowing the functions f_i to take any real value. The problem we have to solve in this case is given by

$$\min_{f \in \mathbb{R}^N} \{f^\top \mathbf{L} f\} \quad s.t. \begin{cases} \mathbf{D} f \perp \mathbf{1} \\ f^\top \mathbf{D} f = \text{vol}(\mathscr{S}). \end{cases}$$

If we substitute function f by $g = \mathbf{D}^{\frac{1}{2}} f$, we arrive at

$$\min_{g \in \mathbb{R}^N} \{g^\top \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}} g\} \quad s.t. \begin{cases} g \perp \mathbf{D}^{\frac{1}{2}} \mathbf{1} \\ \|g\|^2 = \text{vol}(\mathscr{S}). \end{cases}$$

Note that $\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$ corresponds with $\mathbf{L}_{sym}$ defined in Definition 2.3. Applying the Rayleigh–Ritz theorem as in previous approaches, the solution to this problem will be $\mathcal{g}^{\text{op}}$, which is the first eigenvector of $\mathbf{L}_{sym}$ corresponding to the lowest non-zero eigenvalue. If we consider $\tilde{f} = \mathbf{D}^{-\frac{1}{2}}\mathcal{g}$, we obtain the eigenvectors of $\mathbf{L}_{rw}$, as there exists a direct relation between both graph Laplacians, symmetric and random walks ones. Recall that the $\mathbf{L}_{rw}$ eigenvectors are the same that solved the generalized problem $\mathbf{L}\vartheta = \lambda\mathbf{D}\vartheta$. Thus solving the NCut problem, we obtain the same results that we will have achieved with the normalized SC methods shown in Algorithm 2.2 and Algorithm 2.3.

We can extend this explanation to an arbitrary κ, and for this purpose, as done for the extension of the RatioCut problem, we define the vector $\mathbf{h}_i = (h_{1i}, \dots, h_{\mathcal{N}i})$ such that

$$h_{ji} = \begin{cases} \frac{1}{\sqrt{\text{vol}(\mathscr{A}_i)}} & \mathbf{x}^{(j)} \in \mathscr{A}_i \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to see that this vector $\mathbf{h}$ is directly related with the NCut by

$$\begin{aligned} \mathbf{h}_i^\top \mathbf{L}\mathbf{h}_i &= \sum_{\ell,m=1}^{\mathcal{N}} w_{\ell m}(h_{\ell i} - h_{mi})^2 \\ &= \sum_{\substack{\ell \in \mathscr{A}i \\ m \in \bar{\mathscr{A}_i}}} w_{\ell m} \left(\frac{1}{\sqrt{\text{vol}(\mathscr{A}_i)}} \right)^2 + \sum_{\substack{\ell \in \bar{\mathscr{A}_i} \\ m \in \mathscr{A}_i}} w_{\ell m} \left(-\frac{1}{\sqrt{\text{vol}(\mathscr{A}_i)}} \right)^2 \\ &= \frac{1}{\text{vol}(\mathscr{A}_i)} \left(\sum_{\substack{\ell \in \mathscr{A}_i \\ m \in \bar{\mathscr{A}_i}}} w_{\ell m} + \sum_{\substack{\ell \in \bar{\mathscr{A}_i} \\ m \in \mathscr{A}_i}} w_{\ell m} \right) \\ &= 2\frac{\text{cut}(\mathscr{A}_i, \bar{\mathscr{A}_i})}{\text{vol}(\mathscr{A}_i)} = \text{NCut}(\mathscr{A}_i, \bar{\mathscr{A}_i}), \end{aligned}$$

where the symmetry of $\mathbf{W}$ has been used in the last equality.

In matrix notation,

$$\begin{aligned} \text{NCut}(\mathscr{A}_1, \dots, \mathscr{A}_\kappa) &= \frac{1}{2}\sum_{i=1}^{\kappa} \mathbf{h}_i^\top \mathbf{L}\mathbf{h}_i \\ &= \frac{1}{2}\sum_{i=1}^{\kappa} (\mathbf{H}^\top \mathbf{L}\mathbf{H})_{ii} \\ &= \frac{1}{2}\text{Tr}(\mathbf{H}^\top \mathbf{L}\mathbf{H}). \end{aligned}$$

In this case, the matrix $\mathbf{H}$ satisfies $\mathbf{H}^\top \mathbf{D}\mathbf{H} = \mathbf{I}$. In fact, we have

$$\begin{aligned}
\mathbf{h}_i^\top \mathbf{D}\mathbf{h}_i &= \sum_\ell d_\ell h_{\ell i}^2 \\
&= \sum_{\ell \in \mathscr{A}_i} d_\ell \frac{1}{\text{vol}(\mathscr{A}_i)} \\
&= \frac{1}{\text{vol}(\mathscr{A}_i)} \sum_{\ell \in \mathscr{A}_i} \sum_{m=1}^{N} w_{\ell m} \\
&= 1.
\end{aligned}$$

As a consequence, we can rewrite the minimization problem in the following way,

$$\begin{aligned}
&\min_{\mathscr{A}_1,\ldots,\mathscr{A}_\kappa} \{\text{Tr}(\mathbf{H}^\top \mathbf{L}\mathbf{H})\} \\
&s.t. \begin{cases} \mathbf{H}^\top \mathbf{D}\mathbf{H} = \mathbf{I} \\ h_{ji} = \begin{cases} \frac{1}{\sqrt{\text{vol}(\mathscr{A}_i)}} & \mathbf{x}^{(j)} \in \mathscr{A}_i \\ 0 & \text{otherwise.} \end{cases} \end{cases}
\end{aligned}$$

Now we can relax the problem allowing any real entry in matrix $\mathbf{H}$. Also, we define the new normalized matrix $\mathbf{U}$ as $\mathbf{U} = \mathbf{D}^{\frac{1}{2}}\mathbf{H}$.

$$\begin{aligned}
&\min_{\mathbf{U} \in \mathbb{R}^{N \times \kappa}} \{\text{Tr}(\mathbf{U}^\top \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}} \mathbf{U})\} \\
&s.t.\ \mathbf{U}^\top \mathbf{U} = \mathbf{I}.
\end{aligned}$$

And once more, thanks to the Rayleigh–Ritz theorem, we know that a solution will be a matrix $\mathbf{U}^{\text{op}}$ which has the first κ eigenvectors of $\mathbf{L}_{sym}$ as columns. Defining $\tilde{\mathbf{H}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{U}^{\text{op}}$, the new matrix will be formed by the first κ eigenvectors of $\mathbf{L}_{rw}$ that coincide with the ones that solve the generalized problem $\mathbf{L}\vartheta = \lambda \mathbf{D}\vartheta$.

Thus, we have arrive at the same solutions obtained when applying normalized SC algorithms, as the matrix $\mathbf{U}^{\text{op}}$ is equivalent to the symmetric graph Laplacian eigenvector matrix ϑ after applying κ-means over its rows, and the matrix $\tilde{\mathbf{H}}$ is equivalent to the random walk graph Laplacian eigenvector matrix also after clustering its rows. We have just seen that SC methods are good clustering techniques, as they provide approximate solutions to the graph cut minimization problem.

2.4.2.2 Random Walks Point of View

In this subsection we present another way to justify the SC methods based, in this case, on random walks over a graph. Random walks are stochastic processes that, when defined on graphs, jump in a random way from one vertex to another. We can also explain these processes from a partition point of a view, where we would like that

- the random walk spend more time inside the cluster, and
- the random walk rarely jumps to another cluster.

These types of processes are characterized by a transition probability $\mathbf{P} = \{p_{ij}\}$, that in this case will be defined as

$$p_{ij} = \frac{w_{ij}}{d_i}$$
$$\mathbf{P} = (p_{ij})_{i,j=1,\dots,N} = \mathbf{D}^{-1}\mathbf{W}.$$

If we assume that our graph $\mathscr{G}$ is connected and non-bipartite, i.e. its vertices can not be divided into two disjoint sets $\mathscr{A}$ and $\mathscr{B}$ such that every edge connects a vertex in $\mathscr{A}$ to one in $\mathscr{B}$, then the associated random walk always has a unique stationary distribution, that is the limit distribution of a stochastic process with an initial distribution $\Pi = (\Pi_1, \dots, \Pi_N)^\top$ such that

$$\Pi_i = \frac{d_i}{\text{vol}(\mathscr{G})}.$$

With the previous definition, we show that a graph with a low cut will also have few possibilities of jumping between clusters. Thus, there exists a clear relation between random walks and the NCut problem, as

$$\mathbf{L}_{rw} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W} = \mathbf{I} - \mathbf{P}.$$

Consequently, λ_i is an eigenvalue of $\mathbf{L}_{rw}$ with eigenvector ϕ_i only if $(1-\lambda_i)$ is an eigenvalue of $\mathbf{P}$ with eigenvector ϕ_i. Then, to make both methods equivalent we should just take the largest eigenvalues of $\mathbf{P}$ and the smallest eigenvalues of $\mathbf{L}_{rw}$.

Proposition 2.1 (NCut via Transition Probabilities)**.** *Let $\mathscr{G}$ be a connected and non-bipartite graph. Consider the random walk $(X_t)_{t\geqslant 0}$ that starts in X_0 and has associated the stationary distribution Π. Then,*

$$\textit{NCut}(\mathscr{A}, \bar{\mathscr{A}}) = \mathbf{P}(\bar{\mathscr{A}}|\mathscr{A}) + \mathbf{P}(\mathscr{A}|\bar{\mathscr{A}}),$$

where, for two disjoint subsets, $\mathscr{A}, \mathscr{B} \subset \mathscr{S}$, we can define

$$\mathbf{P}(\mathscr{B}|\mathscr{A}) = \mathbf{P}(X_1 \in \mathscr{B}|X_0 \in \mathscr{A}).$$

Proof. We are going to start computing in general the value of $\mathbf{P}(\mathscr{B}|\mathscr{A})$, for $\mathscr{A}$ and $\mathscr{B}$ disjoint subsets. First, we can compute the probability of being in subset $\mathscr{A}$ at time $t = 0$ and in subset $\mathscr{B}$ at time $t = 1$, that is,

$$\begin{aligned}\mathbf{P}(X_0 \in \mathscr{A}, X_1 \in \mathscr{B}) &= \sum_{\substack{i\in\mathscr{A}\\ j\in B}} \mathbf{P}(X_0 = i, X_1 = j)\\ &= \sum_{\substack{i\in\mathscr{A}\\ j\in\mathscr{B}}} \Pi_i p_{ij}\\ &= \sum_{\substack{i\in\mathscr{A}\\ j\in\mathscr{B}}} \frac{d_i}{\text{vol}(\mathscr{G})}\frac{w_{ij}}{d_i}\\ &= \frac{1}{\text{vol}(\mathscr{G})}\sum_{\substack{i\in\mathscr{A}\\ j\in\mathscr{B}}} w_{ij}.\end{aligned}$$

Using this expressions and the fact that $\mathbf{P}(X_0 \in \mathscr{A}) = \frac{\text{favourable cases}}{\text{possible cases}} = \frac{\text{vol}(\mathscr{A})}{\text{vol}(\mathscr{G})}$, we can compute the conditional probability $\mathbf{P}(X_1 \in \mathscr{B}|X_0 \in \mathscr{A})$ as

$$\begin{aligned}\mathbf{P}(X_1 \in \mathscr{B}|X_0 \in \mathscr{A}) &= \frac{\mathbf{P}(X_0 \in \mathscr{A}, X_1 \in \mathscr{B})}{\mathbf{P}(X_0 \in \mathscr{A})}\\ &= \frac{\frac{1}{\text{vol}(\mathscr{G})}\sum_{\substack{i\in\mathscr{A}\\ j\in\mathscr{B}}} w_{ij}}{\frac{\text{vol}(\mathscr{A})}{\text{vol}(\mathscr{G})}}\\ &= \frac{1}{\text{vol}(\mathscr{A})}\sum_{\substack{i\in\mathscr{A}\\ j\in\mathscr{B}}} w_{ij}.\end{aligned}$$

This result plus equality $\mathbf{P}(\mathscr{B}|\mathscr{A}) = \mathbf{P}(X_1 \in \mathscr{B}|X_0 \in \mathscr{A})$, let us define the conditional probabilities between a set and its complement:

$$\begin{cases}\mathbf{P}(\bar{\mathscr{A}}|\mathscr{A}) = \mathbf{P}(X_1 \in \bar{\mathscr{A}}|X_0 \in \mathscr{A}) = \frac{1}{\text{vol}(\mathscr{A})}\sum_{\substack{i\in\mathscr{A}\\ j\in\bar{\mathscr{A}}}} w_{ij},\\ \mathbf{P}(\mathscr{A}|\bar{\mathscr{A}}) = \mathbf{P}(X_1 \in \mathscr{A}|X_0 \in \bar{\mathscr{A}}) = \frac{1}{\text{vol}(\bar{\mathscr{A}})}\sum_{\substack{i\in\bar{\mathscr{A}}\\ j\in\mathscr{A}}} w_{ij}),\end{cases}$$

and adding both expressions, we obtain the NCut definition:

$$\begin{aligned}\mathbf{P}(\bar{\mathscr{A}}|\mathscr{A}) + \mathbf{P}(\mathscr{A}|\bar{\mathscr{A}}) &= \sum_{\substack{i\in\mathscr{A}\\ j\in\bar{\mathscr{A}}}} w_{ij}\left(\frac{1}{\text{vol}(\mathscr{A})} + \frac{1}{\text{vol}(\bar{\mathscr{A}})}\right)\\ &= \text{NCut}(\mathscr{A}, \bar{\mathscr{A}}).\end{aligned}$$

□

This proposition shows that, when we minimize the NCut problem, we are looking for a graph cut such that a random walk defined on it has few transitions from $\mathscr{A}$ to $\bar{\mathscr{A}}$ and vice versa.

Another way of relating random walks and Graph Laplacians is via the commute distance, as follows.

Definition 2.8 (Commute Distance)**.** We define the *commute distance* (or resistance distance) c_{ij} as the expected time (which is equivalent to the number of steps on a discrete scenario) to travel in the random walk from a vertex i to a vertex j and coming back.

This measure decreases if there exist many short paths that connect i and j, and its main characteristic is that it is focused on the set of shortest paths instead of looking only for the shortest one.

It can be shown that the commute distance is equivalent to the pseudo-inverse of the graph Laplacian $\mathbf{L}^\dagger$. In fact, by decomposing the graph Laplacian as

$$\mathbf{L} = \vartheta \mathbf{\Lambda} \vartheta^\top,$$

its pseudo-inverse can be expressed by

$$\begin{aligned}\mathbf{L}^\dagger &= [\mathbf{L}^\top \mathbf{L}]^{-1}\mathbf{L}^\top = [(\vartheta \mathbf{\Lambda}^\top \vartheta^\top)(\vartheta \mathbf{\Lambda} \vartheta^\top)]^{-1}(\vartheta \mathbf{\Lambda}^\top \vartheta^\top) \\ &= [\vartheta \mathbf{\Lambda}^\top \mathbf{\Lambda} \vartheta^\top]^{-1}(\vartheta \mathbf{\Lambda}^\top \vartheta^\top) = \vartheta \mathbf{\Lambda}^\top \mathbf{\Lambda}^{-1} \vartheta^{-1} \vartheta \mathbf{\Lambda}^\top \vartheta^\top \\ &= \vartheta \mathbf{\Lambda}^\top \mathbf{\Lambda}^{-1} \mathbf{\Lambda}^\top \vartheta^\top = \vartheta \mathbf{\Lambda}^\dagger \vartheta^\top,\end{aligned}$$

where ϑ is the matrix that contains the eigenvectors of $\mathbf{L}$ and $\mathbf{\Lambda}^\dagger$ is a diagonal matrix with entries $1/\lambda_i$ when $\lambda_i \neq 0$, and 0 when $\lambda_i = 0$. Therefore, each pseudo-inverse Laplacian matrix element is given by

$$l_{ij}^\dagger = \sum_{m=1}^{N} \lambda_m^\dagger \phi_{im} \phi_{jm}.$$

Proposition 2.2 (Commute Distance Value)**.** *Let $\mathscr{G}$ be a connected and undirected graph, c_{ij} the commute distance between i and j, and $\mathbf{L}^\dagger = (l_{ij}^\dagger)_{i,j=1,\dots,N}$ the pseudo-inverse of $\mathbf{L}$. Then,*

$$\begin{aligned}c_{ij} &= vol(\mathscr{G})(l_{ii}^\dagger - 2l_{ij}^\dagger + l_{jj}^\dagger) \\ &= vol(\mathscr{G})(\mathbf{e}_i - \mathbf{e}_j)^\top \mathbf{L}^\dagger (\mathbf{e}_i - \mathbf{e}_j),\end{aligned}$$

where $\mathbf{e}_i$ is defined as the i^{th} column of the identity matrix $\mathbf{I}$.

The proof of this proposition can be found at Fouss et al. [2007]. This result implies that $\sqrt{c_{ij}}$ can be considered a distance between the graph's vertices induced by the inner product defined below. This means that we can construct an embedding $\mathbf{x}^{(i)} \in \mathscr{S} \to \mathbf{y}^{(i)} \in \mathbb{R}^M$ such that the Euclidean distance between the points $\mathbf{y}^{(i)}$ matched up with the commute distance between the corresponding points on the graph. The embedding is built following these steps:

1. $\mathbf{L}^\dagger$ is a positive semidefinite matrix, so it induces an inner product on $\mathbb{R}^M$.
2. We take now the points $\mathbf{y}^{(i)} \in \mathbb{R}^M$ as the rows of $(\mathbf{\Lambda}^\dagger)^{\frac{1}{2}}\vartheta$.
3. At the end, applying Proposition 2.2 and constructing the matrix $\mathbf{L}^\dagger$ we obtain by construction
$$< \mathbf{y}^{(i)}, \mathbf{y}^{(j)} >= \mathbf{e}_i^\top \mathbf{L}^\dagger \mathbf{e}_j,$$
which directly implies
$$\|\mathbf{y}^{(i)} - \mathbf{y}^{(j)}\|^2 =< \mathbf{y}^{(i)} - \mathbf{y}^{(j)}, \mathbf{y}^{(i)} - \mathbf{y}^{(j)} >= c_{ij}.$$

Even though SC and this commute distance procedure are very similar and both try to construct clusters based on a Euclidean distance between points in the constructed embedding, observe that there exist some important differences:

- In SC the points $\mathbf{x}^{(i)}$ of the graph are embedded into the points $\mathbf{y}^{(i)}$, which are formed by the rows of ϑ. In the commute distance, the points $\mathbf{x}^{(i)}$ are embedded into the points $\mathbf{y}^{(i)}$, by assigning to each point the corresponding row of $(\lambda^\dagger)^{\frac{1}{2}}\vartheta$, where ϑ is the matrix formed by the eigenvectors of $\mathbf{L}$.
- With the commute distance, the entries of $\mathbf{y}^{(i)}$ are scaled by $\frac{1}{\lambda_i}$.
- To obtain the embedding using SC methods, we take just the first κ columns of ϑ, but in the case of the commute distance we use all the columns of the matrix.

2.4.3 Graph Laplacian Comparison and Conclusions

In this section we have introduced the SC algorithms, that present many advantages as they are simple algorithms which can be implemented in a simple way just using standard linear algebra methods, without regarding the data sets size. Also, they do not make any assumption about the cluster's form, so they can solve very general problems as intertwined spirals, as they take into account the geometric information of local data.

These algorithms can be a very powerful tool if we apply them with care. Actually, we should remember that we have to consider several parameters as the selection of a good similarity graph or a convenient number of clusters, which should be carefully chosen if we want to obtain good results. Also note that if we want to update our model with new, unseen data, we would have in principle to repeat the whole process each time we have a new point in our data set.

Other possible problem will be the selection of the most appropriate type of graph Laplacian according to the problem that we want to solve. Note that if we have a regular graph with all

their vertices more or less balanced, all the Laplacians present a similar behavior. However, if the data distribution is not uniform, each different graph Laplacian will result in a different performance.

In general, normalized graph Laplacians outperforms the unnormalized version. This behavior can be explained on one hand taking into account the graph cut principles and on the other hand for consistency reasons.

With respect to the graph cut principles, recall that to solve the RatioCut problem is equivalent to apply the unnormalized SC algorithm, while the NCut problem is directly related with the normalized SC. As explained at the beginning of Section 2.4.2.1, a good clustering method would try to minimize the similarity between clusters while maximizing the similarity inside each cluster. Both methods try to minimize the inter-cluster distance, but only in the NCut problem we try to maximize the intra-cluster distance at the same time. Mathematically, we can see that

$$\sum_{i,j\in\mathscr{A}} w_{ij} = \sum_{\substack{i\in\mathscr{A}\\ j\in\mathscr{A}\cup\bar{\mathscr{A}}}} w_{ij} - \sum_{\substack{i\in\mathscr{A}\\ j\in\bar{\mathscr{A}}}} w_{ij} = \text{vol}(\mathscr{A}) - \text{cut}(\mathscr{A},\bar{\mathscr{A}}).$$

RatioCut does not take into account the intra-class similarity, as it is focused on maximizing the number of nodes $|\mathscr{A}_i|$ and $|\bar{\mathscr{A}_i}|$.

Turning our attention to convergence properties, it can be said that in smooth conditions, both normalized algorithms are consistent from a statistical point of view [Luxburg et al., 2008]. This means that if we assume random data following any distribution with a number of elements which tend to ∞, the results of the normalized SC converge and the limit partition is a sensible partition of the underlying space. This does not work with unnormalized algorithms, as the algorithm could not converge or it does it into trivial solutions. If we want to avoid trivial solutions, we should impose the $\mathbf{L}$ eigenvalues λ_i to be significant under the minimum degree of the graph. As mentioned in [Luxburg et al., 2008], this can be done imposing the constraint

$$\lambda_i \ll \min_{j=1,\dots,N}\{d_j\} \quad i = 1,\dots,\kappa.$$

Between $\mathbf{L}_{rw}$ and $\mathbf{L}_{sym}$, in principle, there does not exist any significant difference. Nevertheless, if we are looking for the intrinsic geometry of the data, the definition of a random walk over the data graph provides interesting properties that, among other things, allow to define another, more general and stable framework, namely the Diffusion Maps (DM), that we will introduce in Chapter 3. We will also see that this new framework is closely related to the previously explained commute distance point of view.

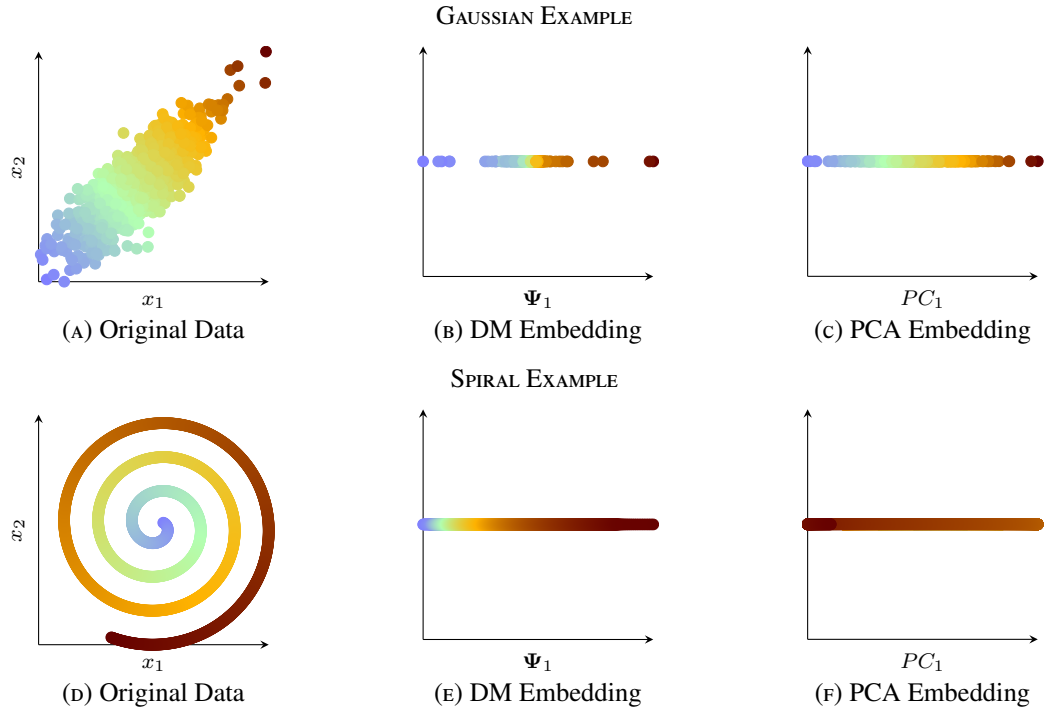
Chapter 3

Diffusion Maps

3.1 Introduction

Methods for dimensionality reduction and data compression, visualization, and analysis that preserve the original information of high dimensional patterns are highly valued in data mining and machine learning. The classical example is Principal Component Analysis (PCA) [Jolliffe, 2002]; [Bishop, 2006, Chapter 12] but in recent years, methods based on manifold learning such as Multidimensional Scaling [Cox and Cox, 2000; Kruskal and Wish, 1978], Locally Linear Embedding [Roweis and Saul, 2000; Saul and Roweis, 2003], Isomap [Seung and Lee, 2000; Tenenbaum et al., 2000], Laplacian Eigenmaps [Belkin and Niyogi, 2001; Belkin and Nyogi, 2003] or Hessian Eigenmaps [Donoho and Grimes, 2003] have received a great deal of attention.

The common assumption in these methods is that sample data lie in a low dimensional manifold, and their goal is to identify the metric on the underlying manifolds, from which a suitable low dimensional representation is derived, that allows to adequately approximate the original but unknown manifold metric with an explicit one in the low dimensional representation. Several of these methods rely on the spectral analysis of a data similarity matrix and this is also the case of Diffusion Maps (DM) [Coifman and Hirn, 2013; Coifman and Lafon, 2006a; Coifman et al., 2005; Nadler et al., 2005].



(A) Original Data (B) DM Embedding (C) PCA Embedding

(D) Original Data (E) DM Embedding (F) PCA Embedding

Figure 3.1: DM and PCA embeddings for a Gaussian sample and a spiral sample.

The goal of DM is to search for a new representation that will capture the main structure of the data in a few dimensions while preserving the local structure of the original data. Since the original data points do not lie on a linear manifold, DM is a nonlinear method.

As it is the case in other manifold learning methods, and understanding geometry as a set of rules describing the relationships between data points, DM relies on a graph representation of the sample giving to the data a geometric notation. This approach was pioneered in the various approaches to Spectral Clustering (SC) [Ng et al., 2001; Shi and Malik, 2000] (see also the review in Luxburg [2007]) and their various but essentially equivalent eigenanalysis of the similarity matrix that, in turn, can be connected with the Riemannian geometry of the manifold where the sample is assumed to lie. The main assumption in DM is that the manifold metric can be approximated by the diffusion distance of a Markov process [Rogers and Williams, 2000a, Chapter 3] whose transition matrix is defined by an adequate normalization of the similarity matrix [Coifman and Lafon, 2006a]. Following on, this allows the construction of a set of embedding functions, the Diffusion Maps, that transform the original space into a new one in which Euclidean distance corresponds to the original manifold metric of the sample data.

All these characteristics let us present this method as a good alternative to classical methods such as PCA, especially when the data present a complex structure. Figure 3.1 shows two synthetic examples. The first example corresponds to a rotated Gaussian centered around zero, while the second is just an equally spaced two-dimensional spiral.

The DM and PCA embeddings obtained for a Gaussian sample are depicted on the top row of the figure. It can be observed that, in this case, both methods are able to find the geometric structure and data order, but it should be pointed out how DM presents a more compact structure for the center of the cloud, where there should be more points, and how far apart from them appear the points which belong to the tail distribution, which could be considered as outliers. In the second example, shown at the bottom row of the figure, the data have a more complex structure that methods like PCA are not able to capture, while DM can perfectly unroll it.

DM is an useful technique to seek a dimensionality reduction that preserves quantities of interest such as local mutual distances or that extracts features to gain insight and understanding about the phenomena that generate the data and about its meaningful structures.

Dimensionality reduction and clustering methods are most often applied in an unsupervised setting, but even if the ultimate goal is to build a supervised classifier or a predictive model, the first objective will be to acquire a knowledge of the underlying data that may be simply impossible to achieve under their original, high dimensionality representations. This is a common situation in many of the modern applications of big data analytics that have to deal with large samples whose also large dimension makes very difficult and even precludes the meaningful use of plain data understanding or visualization tools.

3.2 Theoretical Background

3.2.1 Diffusion Processes

In a physical context, *diffusion* is the process by which a gas moves from regions of high density to regions of lower density according to the relative pressure of each region. This concept can be adapted for example in a graph context in such a way that the diffusion will be a model of spread across the graph. We can think, for example, in the spread of an idea in a social network or the spread of a disease across some region [Newman, 2010, Chapter 6.13.1].

Suppose we have the data organized in an undirected weighted graph, with a_{ij} the similarity between vertex i and j, so the degree of each vertex is $d_i = \sum_j a_{ij}$. Let be χ_i a fluid or substance located in the nodes of the graph that flows from vertex j to an adjacent vertex i with a rate $c(\chi_j - \chi_i)$, where c is a constant usually called *diffusion constant*.

In a short period of time, the flow is given by the quantity $c(\chi_j - \chi_i)dt$, so χ_i changes in a ratio:

$$\begin{aligned}\frac{\partial \chi_i}{\partial t} &= c\sum_j a_{ij}(\chi_j - \chi_i) \\ &= c\sum_j a_{ij}\chi_j - c\chi_i \sum_j a_{ij} \\ &= c\sum_j a_{ij}\chi_j - c\chi_i d_i \\ &= c\sum_j (a_{ij} - \delta_{ij}d_i)\chi_j,\end{aligned}$$

where δ_{ij} is the Kronecker delta. In a matrix notation, this expression is equivalent to

$$\frac{d\chi}{dt} = c(\mathbf{A} - \mathbf{D})\chi.$$

Let $\mathbf{L}$ be the matrix subtraction $\mathbf{L} = \mathbf{D} - \mathbf{A}$. Thus, we arrive to the equation

$$\frac{\partial \chi}{\partial t} + c\mathbf{L}\chi = 0, \tag{3.1}$$

which, if we change the matrix $\mathbf{L}$ by the Laplacian operator ∇^2, is exactly the same equation that the one for a gas diffusion process. As $\mathbf{L}$ plays the role of the Laplacian operator it is usually called the *graph Laplacian*.

This equation can be easily solved taking into account the eigendecomposition of the graph Laplacian matrix $\mathbf{L}$, i.e., considering its eigenvalues and eigenvectors

$$\mathbf{L}\phi_i = \lambda_i \phi_i. \tag{3.2}$$

The eigenvectors $\{\phi_i\}$ form an orthonormal basis, letting us write χ in terms of ϕ_i:

$$\chi(t) = \sum_i w_i(t)\phi_i,$$

being $w_i(t)$ the coefficients to define χ depending on time t.

Replacing this expression in the diffusion equation (Equation (3.1)) and using the eigendecomposition in Equation (3.2) we obtain

$$\frac{\partial(\sum_i w_i(t)\phi_i)}{\partial t} + c\mathbf{L}\sum_i w_i(t)\phi_i = \sum_i \left(\frac{\partial w_i(t)}{\partial t} + c\lambda_i w_i(t)\right)\phi_i = 0.$$

Since the eigenvectors define an orthonormal basis, we arrive to

$$\frac{\partial w_i(t)}{dt} + c\lambda_i w_i(t) = 0,$$

TABLE 3.1: Continuous-Discrete Dictionary.

Continuous version		**Discrete version**	
$\mathcal{P}$	operator	$\mathbf{P}$	matrix
f	function	$\mathbf{v}$	vector
$\phi_\ell(\mathbf{x}^{(i)})$	eigenfunction	$(\phi_\ell)_i$	eigenvector
$\int$	integral	$\sum$	summation

which have as solution:

$$w_i(t) = w_i(0)e^{c\lambda_i t}.$$

In this way, the flux can be determined under some initial conditions $w_i(0)$, just computing the eigendecomposition of the graph Laplacian matrix. We will consider next a particular type of diffusion process that can be used to study the underlying relationship between points in a data set.

3.2.2 Defining Diffusion Coordinates

Even though the study of this theory in a continuous setting is very interesting, especially from an analysis perspective, and it has been the classical way to address it (see [Coifman and Lafon, 2006a; Coifman et al., 2005; Lafon, 2004]), in this study we will suppose that our sample is finite and, accordingly, we will work with its discrete version. In any case, the dictionary in Table 3.1 can be used when translating statements and proofs from one setting to the other.

Let $\mathscr{S} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$ be the starting sample. To define a diffusion process over the data requires to define a random walk over a graph. Thus, the first step is to build a symmetric weighted graph. The weights are given by an affinity measure between the sample points, which is usually given in terms of a kernel matrix $\mathbf{K}$ which results from a kernel operator $\mathcal{K} : \mathscr{S} \times \mathscr{S} \to \mathbb{R}$, defining the matrix entries as $k_{ij} = \mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$. In this chapter we will work in a discrete setting, making occasionally use of the kernel operator point of view. We require it to have the following properties:

- Symmetric: $k_{ij} = k_{ji}$.
- Positive: $k_{ij} \geq 0$.
- Positive semi-definite: $\mathbf{x}^{(i)\top}\mathbf{K}\mathbf{x}^{(i)} \geq 0 \quad \forall \mathbf{x}^{(i)} \in \mathscr{S}$.

Concrete choices for the kernel depend on the concrete problem that we want to solve, but an usual choice is the Gaussian kernel, defined as $k_{ij} = e^{\frac{-\|\mathbf{x}^{(i)}-\mathbf{x}^{(j)}\|^2}{2\sigma^2}}$. In this case we have $k_{ij} > 0$ and the graph is trivially connected.

Once the graph is defined, we show next how to build a Markov Chain (MC) on it. This kind of stochastic process, also called a random walk over a graph, is interesting because it defines a local relationship between points in the graph providing a structure to the data.

Several approaches can be used for this. We will work with the so called *Normalized Graph Laplacian construction* [Chung, 1997], in which we define a transition probability from the affinity matrix. The first step is to compute the degree of each node in the graph as:

$$d_i = \sum_{j \in \mathscr{S}} k_{ij}.$$

Then the transition probability is defined as the normalized kernel:

$$p_{ij} = \frac{k_{ij}}{d_i}. \tag{3.3}$$

This quantity is a well-defined transition probability of the random walk over the data because, on one side, the matrix $\mathbf{P}$ is positive since the $\mathbf{K}$ is positive, and on the other side the sum over all values in the sample space is equal to 1:

$$\sum_j p_{ij} = \frac{\sum_j k_{ij}}{d_i} = \frac{d_i}{d_i} = 1.$$

This transition probability represents the probability of arriving from i to j in one step.

In general, the probability of arriving from i to j in τ steps can be computed as the power $\mathbf{P}^\tau$ of the transition probability matrix and its elements are denoted as p_{ij}^τ.

Note that the parameter τ is giving the scale, which means that running the process far away in time (with a higher τ) lets us integrate the local geometry, so the structure of the data is revealed at different scales (see Coifman et al. [2005] for more details).

This probability lets us talk about a MC over the graph. We can obtain the stationary distribution of the MC as $\Pi_i = \frac{d_i}{\sum_k d_k}$ [Aldous and Fill, 2002, Chapter 3.2]. It is easy to see that Π is indeed stationary given that it satisfies $\Pi\mathbf{P} = \Pi$:

$$\sum_i \Pi_i p_{ij} = \sum_i \frac{d_i}{\sum_k d_k} \frac{k_{ij}}{d_i} = \sum_i \frac{k_{ij}}{\sum_k d_k} = \frac{d_j}{\sum_k d_k} = \Pi_j.$$

Moreover, it satisfies the detailed balance equations:

$$\Pi_i p_{ij} = \frac{d_i}{\sum d_k}\frac{k_{ij}}{d_i} = \frac{k_{ij}}{\sum d_k} = \frac{k_{ji}}{\sum d_k} = \frac{d_j}{\sum d_k}\frac{k_{ji}}{d_j} = \Pi_j p_{ji},$$

being, therefore, a reversible MC [Grimmett and Stirzaker, 2001, Chapter 6.5].

In a less formal way, $\Pi_i p_{ij}$ of the diffusion process can be thought of as the probability flux from state i to state j. Thus the detailed balance equations say that the system is in equilibrium, as there exists a 'local balance' in the sense that the amount flowing from i to j equals the amount flowing from j to i [Grimmett and Stirzaker, 2001, Chapter 6.5].

Since the graph is connected, the chain is irreducible, i.e., it is possible to reach any state from any other state in a finite number of steps. In addition, the chain is aperiodic as for every state i it is satisfied $g(\mathbf{x}^{(i)}) = \gcd\{\tau \geq 1 : p_{ii}^{\tau} > 0\} = 1$. This is true because $p_{ii} = \frac{k_{ii}}{d_i} > 0$, as $k_{ii} > 0 \ \forall i$, arriving at $g(\mathbf{x}^{(i)}) = 1 \quad \forall i$. Thence, the MC is said to be ergodic [Grimmett and Stirzaker, 2001, Chapter 6.2], which means that any state can be reached from any other state in exactly τ steps.

The final objective of this approach is to describe properly the geometry of the data, which means that we would like to find a good distance in the original sample space that characterized well the relation between the points. Moreover, we want to find a dimensionality reduction embedding in which the sample metric is reduced to a simpler one Euclidean distance while retaining the local geometry of the original space.

If we assume that the preceding Markov structure reflects the local aspects of the original sample, a good candidate for a distance that reflects that data structure can be given by

$$\begin{aligned}
\mathcal{D}_\tau(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \|p_{i\cdot}^{\tau} - p_{j\cdot}^{\tau}\|_{(1/\Pi)}^2 &= \sum_k \frac{\left(p_{ik}^{\tau} - p_{jk}^{\tau}\right)^2}{\Pi_k} \\
&= \sum_k \frac{(p_{ik}^{\tau})^2 + (p_{jk}^{\tau})^2 - 2p_{ik}^{\tau}p_{jk}^{\tau}}{\Pi_k} \\
&= \sum_k \frac{p_{ik}^{\tau}p_{ik}^{\tau} + p_{jk}^{\tau}p_{jk}^{\tau} - p_{ik}^{\tau}p_{jk}^{\tau} - p_{jk}^{\tau}p_{ik}^{\tau}}{\Pi_k} \\
&= \sum_k \frac{1}{\Pi_k}\left(p_{ik}^{\tau}p_{ki}^{\tau}\frac{\Pi_k}{\Pi_i} + p_{jk}^{\tau}p_{kj}^{\tau}\frac{\Pi_k}{\Pi_j} - p_{ik}^{\tau}p_{kj}^{\tau}\frac{\Pi_k}{\Pi_j} - p_{jk}^{\tau}p_{ki}^{\tau}\frac{\Pi_k}{\Pi_i}\right) \qquad (3.4) \\
&= \frac{1}{\Pi_i}\sum_k p_{ik}^{\tau}p_{ki}^{\tau} + \frac{1}{\Pi_j}\sum_k p_{jk}^{\tau}p_{kj}^{\tau} - \frac{1}{\Pi_j}\sum_k p_{ik}^{\tau}p_{kj}^{\tau} - \frac{1}{\Pi_i}\sum_k p_{jk}^{\tau}p_{ki}^{\tau} \\
&= \frac{p_{ii}^{2\tau} - p_{ji}^{2\tau}}{\Pi_i} + \frac{p_{jj}^{2\tau} - p_{ij}^{2\tau}}{\Pi_j}, \qquad (3.5)
\end{aligned}$$

where we have applied the detailed balance equations to arrive to Equation (3.4) and for getting Equation (3.5) we have used the Chapman-Kolmogorov equations [Rogers and Williams, 2000a,

Chapter 3.1]. This shows $\mathcal{D}_\tau(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ to be symmetric and we will see below that it satisfies the triangle inequality. Consequently, it is a semi-metric for general kernels, that becomes a metric when the kernel is strictly positive, which is the case with the Gaussian kernel. We thus call it *diffusion distance* as it measures the connectivity between two points in the data set after 2τ steps.

Intuitively, $\mathcal{D}_\tau(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ will be small if there exist a lot of paths that connect $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ i.e., when the transition probabilities $p_{ij}^{2\tau}$ and $p_{ji}^{2\tau}$ are high. Another good property is that it is robust to noise because it is calculated as an average of all the paths of length τ.

The direct computation of this diffusion distance is very expensive computationally as τ grows. To avoid this computational effort, a typical solution is the use of spectral theory instead of computing the matrix powers. The starting point is Theorem 3.1 [Horn and Johnson, 1985, Theorem 4.1.5] that applies to symmetric matrices.

Theorem 3.1 (Spectral Theorem)**.** *Any symmetric matrix $\mathbf{A} \in \mathbb{R}^n \times \mathbb{R}^n$ can be reduced to an orthonormal basis determined by a diagonal matrix, i.e., $\mathbf{A} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}'$, where*

- $\mathbf{\Lambda} = diag\{\lambda_0, \lambda_1, \cdots, \lambda_N\}$*, with λ_ℓ the eigenvalues of* $\mathbf{A}$*,*
- $\mathbf{\Phi} = (\phi_0, \cdots, \phi_N)$*, with ϕ_ℓ the eigenvectors of* $\mathbf{A}$*.*

Since our Markov matrix is not symmetric, we will consider first its conjugated matrix $\mathbf{A}$ defined as

$$a_{ij} = \frac{\sqrt{\Pi_i}}{\sqrt{\Pi_j}} p_{ij} = \frac{\sqrt{\Pi_i}}{\sqrt{\Pi_j}} \frac{k_{ij}}{d_i} = \frac{\sqrt{\frac{d_i}{\sum d_k}}}{\sqrt{\frac{d_j}{\sum d_k}}} \frac{k_{ij}}{d_i} = \frac{k_{ij}}{\sqrt{d_i}\sqrt{d_j}},$$

which is symmetric.

Considering the symmetry of matrix $\mathbf{A}$, Theorem 3.1 implies that $\mathbf{A}$ has a discrete set of eigenvalues $\{\lambda_\ell\}_{\ell \geq 0}$ and it satisfies that

$$a_{ij} = \sum_{\ell \geq 0} \lambda_\ell (\phi_\ell)_i (\phi_\ell)_j,$$

where $\{\phi_\ell\}_{\ell \geq 0}$ is an orthonormal set of eigenvectors.

The first eigenvalue is always $\lambda_0 = 1$ as it corresponds to the eigenvector $\phi_0 = \sqrt{\Pi}$, for which we have

$$(\mathbf{A}\sqrt{\Pi})_i = \sqrt{\Pi_i}.$$

In fact, it can be shown that

$$(\mathbf{A}\sqrt{\Pi})_i = \sum_j a_{ij}\sqrt{\Pi_j} = \sum_j \frac{\sqrt{\Pi_i}}{\sqrt{\Pi_j}} p_{ij} \sqrt{\Pi_j} = \sqrt{\Pi_i} \sum_j p_{ij} = \sqrt{\Pi_i}\mathbf{1} = \sqrt{\Pi_i} \quad \forall i.$$

Moreover, we can write the transition probability matrix $\mathbf{P}$ as

$$p_{ij} = \frac{\sqrt{\Pi_j}}{\sqrt{\Pi_i}} a(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \frac{\sqrt{\Pi_j}}{\sqrt{\Pi_i}} \sum_{\ell \geq 0} \lambda_\ell (\phi_\ell)_i (\phi_\ell)_j = \sum_{\ell \geq 0} \lambda_\ell \frac{(\phi_\ell)_i}{\sqrt{\Pi_i}} (\phi_\ell)_j \sqrt{\Pi_j},$$

or, in other words,

$$p_{ij} = \sum_{\ell \geq 0} \lambda_\ell (\psi_\ell)_i (\varphi_\ell)_j,$$

where we define $(\psi_\ell)_i = \frac{(\phi_\ell)_i}{\sqrt{\Pi_i}}$ and $(\varphi_\ell)_j = (\phi_\ell)_j \sqrt{\Pi_j}$. Moreover, we can argue again that $\lambda_0 = 1$ as $\sum_j p_{ij} (\psi_0)_i = \lambda_0 (\psi_0)_i$.

In general, the number of connected components of a graph coincide with the multiplicity of $\lambda_\ell = 1$ [Newman, 2010, Chapter 6.13.3]. In our case, the graph is connected and there is just one $\lambda_0 = 1$ eigenvalue. The other eigenvalues will satisfy $|\lambda_\ell| < 1 \; \forall \ell \geq 1$ [Aldous and Fill, 2002, Chapter 3.4].

Now it is easy to see that the eigendecomposition of $\mathbf{P}$ can be expressed for any τ-steps as:

$$p_{ij}^\tau = \sum_{\ell \geq 0} \lambda_\ell^\tau (\psi_\ell)_i (\varphi_\ell)_j.$$

This can be exploited to simplify the previously defined diffusion distance $\mathcal{D}_\tau(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$. In fact, we have

$$\begin{aligned}
\mathcal{D}_\tau(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) &= \|p_{ik}^\tau - p_{jk}^\tau\|^2_{(1/\Pi)} \\
&= \sum_k \frac{\left(p_{ik}^\tau - p_{jk}^\tau\right)^2}{\Pi_k} \\
&= \sum_k \frac{\left(\sum_{\ell \geq 0} \lambda_\ell^\tau (\psi_\ell)_i (\varphi_\ell)_k - \sum_{\ell \geq 0} \lambda_\ell^\tau (\psi_\ell)_j (\varphi_\ell)_k\right)^2}{\Pi_k} \\
&= \sum_k \frac{\sum_{\ell \geq 0} \lambda_\ell^{2\tau} \left((\psi_\ell)_i - (\psi_\ell)_j\right)^2 (\varphi_\ell)_k^2}{\Pi_k} \qquad (3.6) \\
&= \sum_{\ell \geq 0} \lambda_\ell^{2\tau} \left((\psi_\ell)_i - (\psi_\ell)_j\right)^2 \sum_k \frac{(\varphi_\ell)_k^2}{\Pi_k} \\
&= \sum_{\ell \geq 0} \lambda_\ell^{2\tau} \left((\psi_\ell)_i - (\psi_\ell)_j\right)^2 \sum_k \frac{((\phi_\ell)_k \sqrt{\Pi_k})^2}{\Pi_k} \qquad (3.7) \\
&= \sum_{\ell \geq 0} \lambda_\ell^2 \left((\psi_\ell)_i - (\psi_\ell)_j\right)^2,
\end{aligned}$$

where we have applied in Equation (3.6) that the eigenvectors of matrix $\mathbf{P}$ are orthogonal and in Equation (3.7) that the eigenvectors of matrix $\mathbf{A}$ have unitary norm. Notice that this implies that $\mathcal{D}_\tau(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ satisfies the triangle inequality and is thus a distance. Moreover, it is not necessary to take into account all the eigenvalues, as they tend to 0 when ℓ grows, and thus their

contribution to the diffusion distance will be very small. In fact, for a given precision δ, we can work only with the most relevant eigenvalues:

$$\bar{M} = s(\delta, t) = \max\{\ell \in \mathbb{N} \text{ s.t. } |\lambda_\ell|^t > \delta|\lambda_1|^t\}.$$

Even more, the term with $\ell = 0$ can be omitted, because, as we have proved, its eigenvector ψ_0 is constant, which means that it collapses all the elements of each point onto the real number 1, which does not give any information. Thus, we ignore the first eigenvalue.

With the above choices, the diffusion distance can be approximated in a possibly much lower dimensional space as

$$\mathcal{D}_t(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \approx \sum_{\ell \geq 1}^{\bar{M}} \lambda_\ell^{2t} \left((\psi_\ell)_i - (\psi_\ell)_j\right)^2.$$

If we want to embed the data, the natural coordinates will be defined by

$$\mathbf{\Psi}_t(\mathbf{x}) = \begin{pmatrix} \lambda_1^t \psi_1(\mathbf{x}) \\ \vdots \\ \lambda_{\bar{M}}^t \psi_{\bar{M}}(\mathbf{x}) \end{pmatrix}.$$

These coordinates are called *Diffusion Coordinates*, and we call *Diffusion Maps* to the family $\{\mathbf{\Psi}_t\}_{t \in \mathbb{N}}$. These maps embed the data into the Euclidean space $\mathbb{R}^{\bar{M}}$ in such a way that the Euclidean distance in the embedded space approximates the diffusion distance in the original space, up to the relative precision δ

$$\|\mathbf{\Psi}_t(\mathbf{x}^{(i)}) - \mathbf{\Psi}_t(\mathbf{x}^{(j)})\| = \mathcal{D}_t(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \sum_{\ell \geq 1}^{\bar{M}} \lambda_\ell^{2t} \left((\psi_\ell)_i - (\psi_\ell)_j\right)^2 + \sum_{\ell > \bar{M}}^{N} \lambda_\ell^{2t} \left((\psi_\ell)_i - (\psi_\ell)_j\right)^2,$$

where the second term expresses the approximation error when we just work with $\bar{M}$ dimensions, and it can be bounded by

$$\begin{aligned} \sum_{\ell > \bar{M}}^{N} \lambda_\ell^{2t} \left((\psi_\ell)_i - (\psi_\ell)_j\right)^2 &\leq \sum_{\ell > \bar{M}}^{N} \delta \lambda_1^{2t} \left((\psi_\ell)_i - (\psi_\ell)_j\right)^2 \\ &\leq \delta \lambda_1^{2t} \sum_{\ell > \bar{M}}^{N} \left((\psi_\ell)_i - (\psi_\ell)_j\right)^2 \\ &\leq 4(N - \bar{M})\delta, \end{aligned} \tag{3.8}$$

where in Equation (3.8) we have applied that the absolute value of the components of each eigenvector is bounded by one since they are orthonormal.

After all these steps, we have obtained a new representation of the original points in a Euclidean space of smaller dimension, preserving their geometrical structure.

3.2.3 Density Influence

It is clear that the distribution of the sample data is an important factor that will affect how well the similarity matrix captures the local geometry of the data. It is thus important to take this distribution into account, and following the discussion in Coifman and Lafon [2006a] a new parameter α, with values between 0 and 1 is introduced for making explicit the influence of the sample density, which is measured by the degree of the graph. To do so, we will not work directly with $\mathbf{K}$ nor with $\mathbf{P}$, but instead with the α-dependent density normalized matrix

$$k_{ij}^{(\alpha)} = \frac{k_{ij}}{d_i^{\alpha} d_j^{\alpha}}$$

where $d_i = \sum_{j=1}^{N} k_{ij}$ is the graph degree of each vertex $\mathbf{x}^{(i)}$.

After this pre-normalization, the classical normalization explained in Equation (3.3) is applied over $\mathbf{K}^{(\alpha)}$ to obtain the new transition probability matrix $\mathbf{P}^{(\alpha)}$,

$$p_{ij}^{(\alpha)} = \frac{k_{ij}^{(\alpha)}}{d_i^{(\alpha)}},$$

where $d_i^{(\alpha)} = \sum_{j=1}^{N} k_{ij}^{(\alpha)}$. Notice that we can apply the discussion in Section 3.2.2 to the MC defined by the new $\mathbf{P}^{(\alpha)}$ just as done there.

The remaining question is how to choose the parameter α. It can be seen [Coifman and Lafon, 2006a] that the infinitesimal generator ∇_α of the diffusion process acts on a function f as

$$\nabla_\alpha f = \frac{\Delta(f\mathbf{d}^{1-\alpha})}{\mathbf{d}^{1-\alpha}} - \frac{\Delta(\mathbf{d}^{1-\alpha})}{\mathbf{d}^{1-\alpha}} f,$$

where Δ is the Laplace–Beltrami of the underlying manifold. Notice that if $\alpha = 1$, ∇_1 coincides with Δ and we can expect the diffusion projection to capture the underlying geometry without any interference from the sample's density $\mathbf{d}$. On the other hand, when $\alpha = 0$, we have

$$\nabla_0 f = \frac{\Delta(f\mathbf{d})}{\mathbf{d}} - \frac{\Delta(\mathbf{d})}{\mathbf{d}} f$$

and the density $\mathbf{d}$ will influence how the diffusion coordinates capture the underlying geometry, unless, of course, $\mathbf{d}$ is uniform in which case we arrive again at $\nabla_0 = \Delta$. Because of this we will consider the case $\alpha = 1$ in all the experiments done.

We will follow this approach in the remainder of the thesis when applying DM. All the previous steps are summarized in Algorithm 3.1.

As it is the case in general, DM requires a proper choice of the parameters involved, that could determine the good performing of this technique. As previously mentioned, the usual choice for

ALGORITHM 3.1: Diffusion Maps Algorithm.

Input: $\mathscr{S} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$, the original data set ; Parameters: τ, α, σ, δ ;

Output: $\mathbf{\Psi}_\tau(\mathbf{x})$, the embedded data set ;

1: $k_{ij} = e^{\frac{-\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2}{\sigma^2}}$; ▶ *Gaussian Kernel matrix.*

2: $k_{ij}^{(\alpha)} = \frac{k_{ij}}{d_i^\alpha d_j^\alpha}$, with $d_i = \sum_{j=1}^{N} k_{ij}$; ▶ *Density Normalization.*

3: $p_{ij} = \frac{k_{ij}^{(\alpha)}}{d_i}$, with $d_i = \sum_{\mathscr{S}} k_{ij}^{(\alpha)}$; ▶ *Transition Probability.*

4: Get $\{\lambda_r, \psi_r\}_{r \geqslant 0}$ s.t. $\begin{cases} 1 = \lambda_0 > |\lambda_1| \geqslant \cdots \\ \mathbf{P}\psi_r = \lambda_r \psi_r \end{cases}$; ▶ $\mathbf{P}$ *Eigendecomposition.*

5: $\bar{M} = \max\{\ell \in \mathbb{N}\ s.t.\ |\lambda_\ell|^\tau > \delta |\lambda_1|^\tau\}$; ▶ *Embedding Dimension.*

6: $\mathbf{\Psi}_\tau(\mathbf{x}) = \begin{pmatrix} \lambda_1^\tau \psi_1(\mathbf{x}) \\ \vdots \\ \lambda_{\bar{M}}^\tau \psi_{\bar{M}}(\mathbf{x}) \end{pmatrix}$; ▶ *Diffusion Coordinates.*

the similarity matrix is to use a Gaussian kernel, defined as,

$$k_{ij} = e^{\frac{-\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2}{2\sigma^2}}.$$

The kernel parameter σ is crucial, as it determines the neighborhood influence to define the diffusion coordinates. A sensible choice is to determine its value depending on the distances between the original points. Some authors have adopted this solution like Rabin [2010], who proposes to fix this value as the median of the distances, or Mouysset et al. [2012], whose proposal is based on the maximum distance over the sample set. The problem to solve determines the method for searching this parameter value. Because of this, our proposal is to determine the σ value as a discrete percentile of this distance

$$\sigma = p_\rho(\mathcal{D}_\tau),$$

where ρ represents the distance percentile, taking values between 0 and 1. As mentioned before, the appropriate value for this ρ parameter will depend on the problem to handle, so we will look for optimal values specific to the concrete problems studied.

Other important parameter, especially if we are interested in dimensionality reduction, is the $s(\delta, \tau)$ function that determines the reduced dimensionality. Recall that, following Coifman and Lafon [2006a], for a given τ, we select $\bar{M}$ dimensions on the form

$$\bar{M} = s(\delta, \tau) = \max\{\ell \in \mathbb{N} \text{ s.t. } |\lambda_\ell|^\tau > \delta |\lambda_1|^\tau\}. \tag{3.9}$$

In other words, we retain those eigenvalues whose τ-power is larger than the fraction δ of λ_1^τ. Thus, the selection of the final dimension reduces to the determination of the precision value δ. Although in our experiments we will fix by hand the parameter δ, one could, instead, determine this parameter in terms of the diffusion distance error as presented in Equation (3.8).

The remaining parameters τ and α also depend on the problem to solve. By default, we set $\alpha = 1$ motivated by our desire of avoiding possible effects due to the underlying density. As for τ, the choice will be problem dependent although we usually take $\tau = 1$ and occasionally 2.

3.2.4 Heterogeneous Attributes

A limitation of the DM method is that it implicitly assumes the attributes to be homogeneous. However, real-life data sets are frequently heterogeneous, something that often cannot be handled just by normalizing the data as we can lose some geometric information. In Rabin and Coifman [2012] a method is proposed to adapt DM to work with heterogeneous features just by dealing separately with groups of attributes that are deemed to be homogeneous.

More precisely, assume that we can classify our features in G groups, normally defined based on expert knowledge about the data set. The collection of points for each group will be denoted by $\mathscr{S}_g$. Then, we can split each pattern $\mathbf{x}^{(i)}$ into G new, lower dimensional ones so $\mathbf{x}^{(i)} = \{\mathbf{x}^{(i,g)}\}_{g=1}^{G}$, being $\mathbf{x}^{(i,g)} = \{x_j^{(i)}\}_{j\in\mathscr{S}_g}$. In this way we have divided our data in G homogeneous groups of points.

We now apply DM as described before to each $\mathscr{S}_g$ obtaining G embeddings $\{\mathbf{\Psi}_g\}_{g=1}^{G}$ that capture the geometry associated to each feature subset. These new projections $\mathbf{\Psi}_g$ are given by eigenvalue-eigenvector products that are now being comparable across the embeddings since all the eigenvectors involved have unit norm. We can make eigenvalues also comparable if we re-scale them as $\hat{\lambda}_i^{(g)} = \frac{\lambda_i^{(g)}}{\sum_j \lambda_j^{(g)}}$. Thus the union of the normalized features $\mathbf{V}$ gives a set of homogeneous features that still represent the intrinsic geometry of our original data and we can simply apply DM again to this new data set to get the final lower dimensional embedding. A graphical scheme for this process can be seen in Figure 3.2.

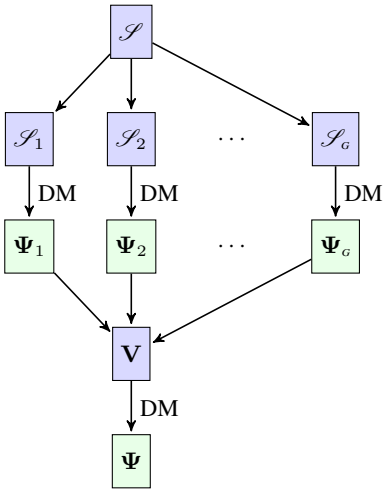


FIGURE 3.2: DM for heterogeneous attributes.

In summary, DM also makes possible low dimensional embeddings of heterogeneous data while transforming the original space metric into a Euclidean one.

3.2.5 Possible Applications of this Theory

Recall that DM is a manifold learning method that searches a new representation which captures the main structure of the data in fewer dimensions, and over the last few years, the use of manifold learning methods to find the intrinsic geometry of a data sample has been successfully applied to a variety of problems like, for example [Coifman and Hirn, 2013], [Xu et al., 2010] or [Bhat and Arnold, 2007]. It is a good method for dimensionality reduction, especially if we are looking to preserve intrinsic geometric characteristics of the data, or to obtain a new representation of the data that reveal their underlying structure.

DM is thus an appropriated method for describing data; it can help to analyze, visualize and to properly organize the data, and also to detect possible anomalies. In this context, it is natural and very helpful to combine DM with clustering methods, something that often yields very good results. We present in Section 3.3 an example where some meteorological data is studied and analyzed using these techniques.

The combination of DM and clustering methods can be also used to identify local regions in the data where local models can then be defined. In Section 3.4 this issue is addressed in the context of wind energy prediction. In contrast, DM may not present big advantages over SC if we use their features to train regression models, primarily due to the fact that DM coordinates are just a re-scaling of the SC embedded coordinates, as they appear multiplied by a power of the eigenvalues.

3.3 DM for Meteorological Data Description

3.3.1 Problem Definition

The analysis of meteorological data is a field that can clearly benefit from dimensionality reduction techniques. Weather forecasts usually involve a large number of variables over large spatial grids: the meteorological agencies provide forecasts for several pressure layers and for several future horizons, so even considering only a single grid point, a weather pattern is often big. For example, the European Center for Medium-Range Weather Forecast (ECMWF) [ECMWF, 2005] offers about 70 different variables, in 25 pressure layers for 75 future horizons. Of course, the selection of the most appropriate data to be used largely depends on the problem at hand, and this fact will determine the dimensionality of the problem to solve.

In this context, we can consider two alternative situations depending on whether we look at patterns as space or time elements. In the first case, we may look a set of forecasts at a single grid point taken over a long time period. In the second, we fix a short time period (say, a day) and consider such a set of forecasts over a large spatial grid. Both problems are natural candidates for data compression, in the first case under a time component and in the second one under a spatial component. Therefore, when dimensionality reduction methods are applied to them we shall talk about *time* and *spatial dimensionality reduction* respectively.

In both cases, large dimensional data vectors are being considered and dimensionality reduction is clearly of interest but, as it is generally the case with unsupervised methods, a reasonable question is how to decide the correctness or usefulness of the new representation. A possible answer can be derived through the use of clustering and to check the relevance of the new coordinates, we will apply κ-means on them and analyze the resulting clusters. As it is well known, it is usually quite difficult to determine the number κ of clusters to be looked for. One option is likelihood-based methods such as Bayesian Information Criterion (BIC) or Integrated Classification Likelihood (ICL) [Biernacki et al., 2000] that assume underlying Gaussian mixture models and penalize their complexity. However, likelihood values are often scale dependent and the complexity penalization may be too small to make up for that. There are other, more robust options such as G-means [Hamerly and Elkan, 2003] that determines in an automatic way the best κ for each problem, assuming Gaussianity in the clustered data, but this is out of the scope of the present thesis. As we shall see there, it usually gives a large number of clusters, something which makes difficult the result visualization. Because of this, here we will fix by hand the number of clusters taking into account the special characteristics of weather data.

In the examples presented in this thesis we apply DM for the analysis of the meteorological data for the Iberian Peninsula. More precisely, we will not work with actual weather measures but, instead, with surface predictions provided by the ECMWF. These predictions are in general close to the actual atmospheric conditions, and they have the advantage of providing values over an uniform large scale grid. To cover the Iberian Peninsula we have selected a $1,995$-point square grid with a resolution of $0.25°$ (i.e., a grid square corresponds to a land square with about a 27Km side).

FIGURE 3.3: Satellite map of Spain.
Source: Jacques Descloitres, MODIS Land Rapid Response Team at NASA Goddard Space Flight Center.

We will use meteorological forecasts for a whole year, from March 2009 to February 2010, working

with five surface variables: wind velocity and direction (decomposed in sine and cosine), pressure and temperature, which are available every three hours. In this way, we have for each day eight snapshots of the meteorological conditions over a large region. Note that, for this kind of study, it is not necessary to divide our data into training and test sets as we are only interested in analyzing the whole set of meteorological data and not in using it for, say, prediction purposes.

3.3.2 Time Compression

For time compression we will consider for each of the $1,995$ nodes in the ECMWF grid for the Iberian Peninsula a vector made up by the 5 variable surface forecasts at that node for the whole year. Thus, recalling that for each day we received 8 snapshots of meteorological data, we assign to each node a vector with dimension $5 \times 8 \times 365 = 14,600$, and we seek to compress that vector into another one of a much lower dimension in a way that still provides meaningful information for each node.

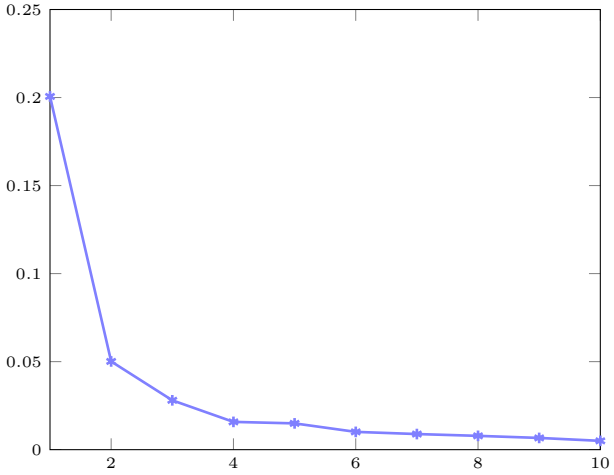
It is logical to expect that points that are close in the reduced space should also be close in the original space. Since we are dealing with surface points, a first notion of closeness could be just geographic proximity but this may be just too narrow, as we would also expect that far away points might share similar weather. We will see with this experiment that the reduced coordinates will give a meaningful representation of the Iberian Peninsula in terms of the climate, for which we will relate it to geographical altitude.

3.3.2.1 Parameter Setting

The first step is always the parameter selection. We will work with Gaussian kernels to build the kernel of the graph defined as

$$k_{ij} = e^{\left(\frac{-\|x_i - x_j\|^2}{2\sigma^2}\right)},$$

and our first task should be to choose the kernel width parameter σ. Recall from Section 3.2 that this parameter defines the neighborhood size. As discussed there, a good idea is fixing it in terms of the Euclidean distance between x_i and x_j for every pair of points in $\mathscr{S}$ [Rabin, 2010]. In this case we have chosen the 50% as percentile, i.e., the median of the sample distances, because this measure is very robust to outliers. For this concrete problem the σ parameter obtained has a value of 159.18. Other distance percentiles have been tested to define σ, but they all provide similar results—probably because in this example all the points appear very concentrated in a region, so to define a more local or global neighborhood does not make a big difference for defining the structure of the data—and we have settled then for the median as a robust, appropriated kernel width parameter.



τ	$\delta = 0.1$	$\delta = 0.01$
1	3	19
2	1	3
3	1	2
4	1	1

FIGURE 3.4: Parameter setting for time compression. The left hand image presents the first 10 eigenvalues (without λ_0) of the probability matrix. The right hand image shows the reduced dimension for some different δ and τ values.

The next choices are the diffusion step τ and the dimension $\bar{m}$ of the reduced space. We first apply the threshold-based dimension selection method proposed in [Coifman and Lafon, 2006a] and already explained in Chapter 3. Recall that for a given τ, we select the dimensionality reduction according to Equation (3.9), where the precision parameter δ is fixed in our experiments either at 0.1 or at 0.01. For instance, when $\tau = 1$ it actually means that we take into account the eigenvalues which are either a 10% or a 1% bigger than the first eigenvalue. The α parameter has been fixed to 1 for all the models.

The right hand table in Figure 3.4 shows $\bar{m}$ values for different values of both δ and τ. The figure at left shows the eigenvalue decay. In that table we discard $\bar{m}$ values of 1, as probably yielding a too drastic dimension reduction. We also discard the probably too high value of 19 obtained for $\delta = 0.01$ and $\tau = 1$. This leaves us with the options $\tau = 1$, $\bar{m} = 3$ for the 10% precision and $\tau = 2$, $\bar{m} = 3$ or $\tau = 3$, $\bar{m} = 2$ for the 1% precision. For a more homogeneous comparison we will work with $\bar{m} = 3$ in both cases. This value also makes sense if we observe the eigenvalue decay, that seems to become linear and too near to 0 around the fourth eigenvalue.

The extreme case $\tau = 0$ represents the classical SC algorithms. In this special case the technique applied to determine the reduced dimension could not be used, as it take into account the τ-th powers of the eigenvalues, which take no part in the classic SC algorithm. For comparison purposes the dimension will be also reduced to three in this case, a choice that, as comment before, is coherent with the eigenvalue curve of the data.

3.3.2.2 Experimental Results

Once we have decided the best parameters for this problem, we can apply DM to it. We will also apply PCA for comparison purposes. In principle, the covariance matrix would be $14,600 \times 14,600$ but since the data matrix has rank at most $1,995$, this is the effective dimension. To make PCA comparable to the diffusion methods with the parameters obtained in the above subsection, we will apply PCA with a three-dimensional projection. These three PC coordinates explain the 44.18% of the variance.

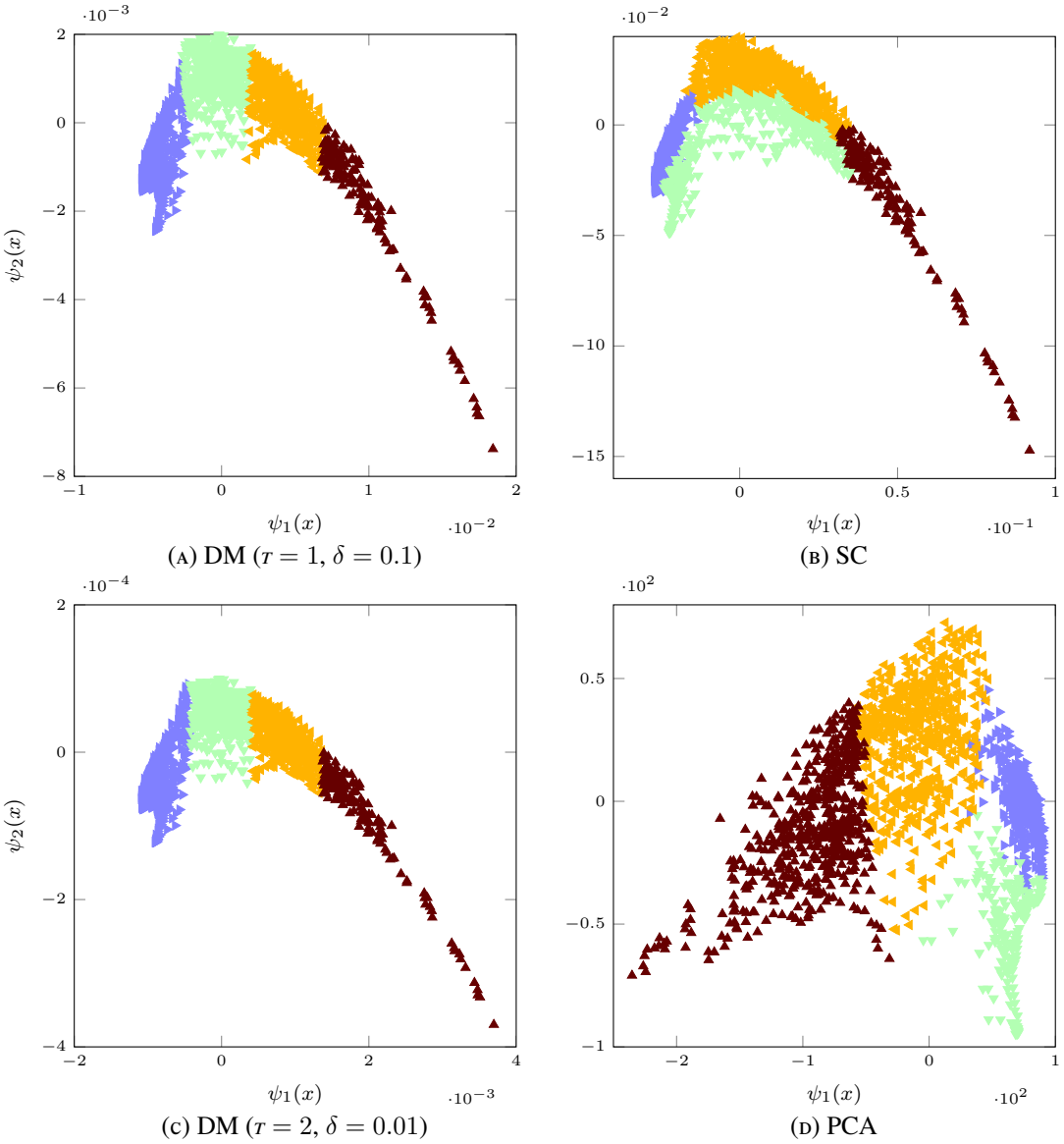
Therefore, we will compare four different models for this experiment: the classical SC algorithm (DM with $\tau = 0$), PCA, and DM with $\tau = 1$ and $\tau = 2$ reducing the dimension coordinates to a three-dimensional space.

If we want to analyze the quality of the embedded coordinates, in addition to observe the structure of the embedding, it is natural to depict clusters over them. In this case we have decided to paint four clusters in order to guarantee a good and useful visualization.

Note that for κ-means clusters the different groups obtained depend on the initialization of its centroids, which are randomly selected. This method is the classical option for centroids initialization and, as Peña et al. [1999] shows, it is robust and efficient but the convergence speed is not the best and different clusters can be obtained for each execution. For these experiments, the random seed has been always the same to guarantee the same results. Anyway, we have tested other random seed initialization and the embeddings and clusters obtained in this case are always very similar.

In Figure 3.5 it can be seen a two-dimensional representation of the corresponding embedding associated to the two largest eigenvalues of each model. DM models present the same embedding structure but at different scales, PCA has a similar structure although it is less clear, more rounded. Regarding to the clusters defined, the different colors are clearly grouped for all the models, although each one has a different distribution.

It is not easy to decide on the quality of these groups just regarding to the embedded coordinates. To do so, we can depict the original ECMWF grid points according to the cluster colors in the embedding. Observing the resulting image in Figure 3.6 it can be concluded that the clusters depict clearly the contour of the Iberian Peninsula as well as that of the north of Africa for most of the models (the real contour has been depicted in black as a reference for an easier visualization). Sea grid points are clearly identified as one of the four clusters for the DM models with $\tau = 1$ and $\tau = 2$, and between these two models we can conclude that $\tau = 1$ is sharper (observe the southwest coast in the case $\tau = 2$ where the sea color come into the peninsula), although they are almost indistinguishable. The other three clusters could be roughly



(A) DM ($\tau = 1$, $\delta = 0.1$) (B) SC

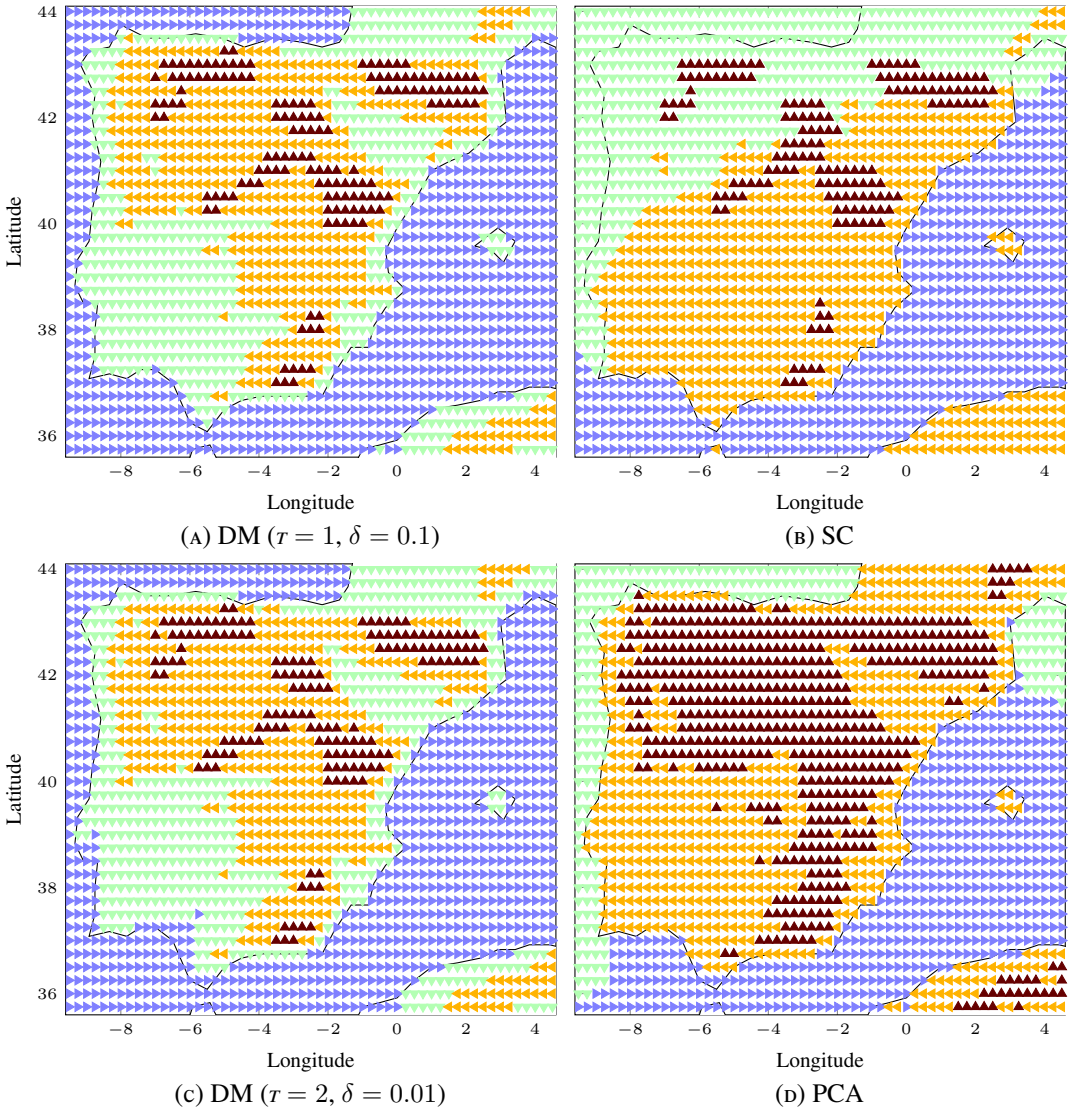
(C) DM ($\tau = 2$, $\delta = 0.01$) (D) PCA

FIGURE 3.5: Embeddings resulting for time compression. The images represent the first two embedded coordinates for the four different models.

assigned to coasts and lower valleys, the central Spain plateaus and the mountains. To verify this, these pictures can be compared with the real satellite image of Spain shown in Figure 3.3.

PCA assigns two clusters for the sea and another two for the land, which hints to a less precise embedded structure. The classical SC model seems to determine well the mountain regions but it somehow separates the North and South halves of the peninsula, giving good clusters in the second one. While it is true that the North has different weather than the South, we think that the climate is more different between different levels than between different regions.

In summary DM dimensionality reduction and clustering of one year of weather prediction from an initial dimension of $14,600$ to a much lower of three gives us a meaningful geographical



(A) DM ($\tau = 1$, $\delta = 0.1$) (B) SC

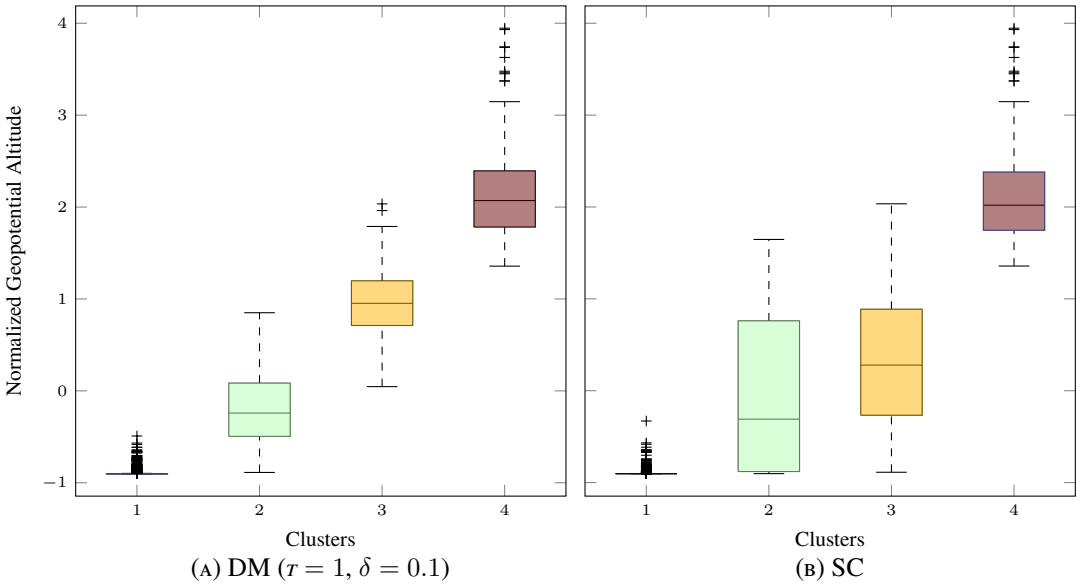
(C) DM ($\tau = 2$, $\delta = 0.01$) (D) PCA

FIGURE 3.6: Maps resulting for time compression.

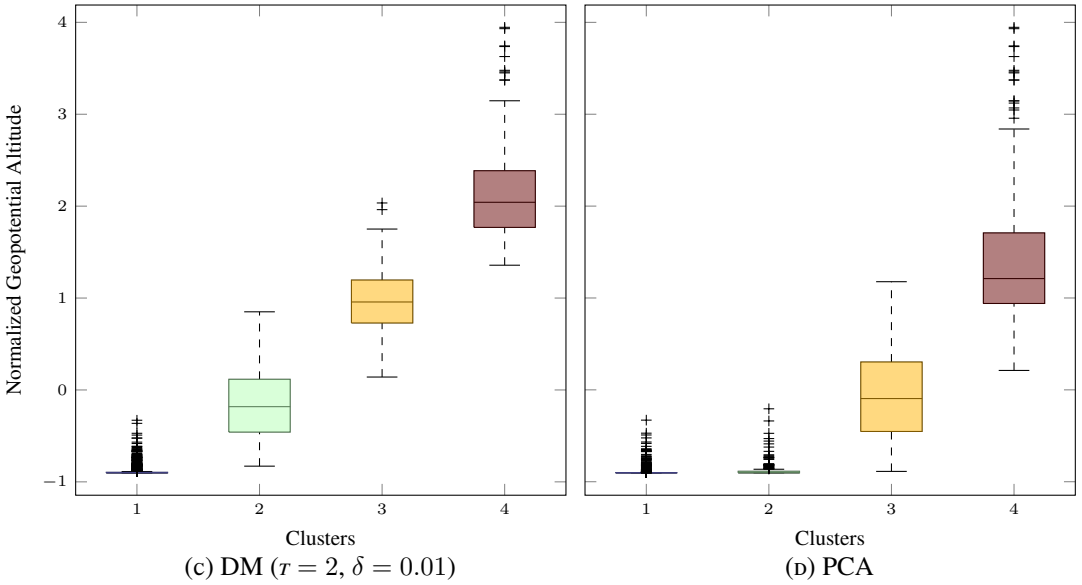
representation of the sample points. The resulting map suggests that the embedding gives what could be a height-based representation of the underlying grid.

Since we work with surface forecasts, each grid point has associated its geopotential in the underlying orography model. We have considered for the four clusters the geopotentials altitude (normalized to zero mean and 1 deviation) of each one of their points. Figure 3.7 depicts geopotential values for each cluster in a box plot diagram. This representation summarizes the sample minimum, the lower quartile, the median, the upper quartile, and the sample maximum. It also indicates which observations, if any, might be considered outliers.

All the points in the sea clusters of the four representations have essentially the same geopotential altitude and that is the reason why just a line is depicted instead of a box for them. In



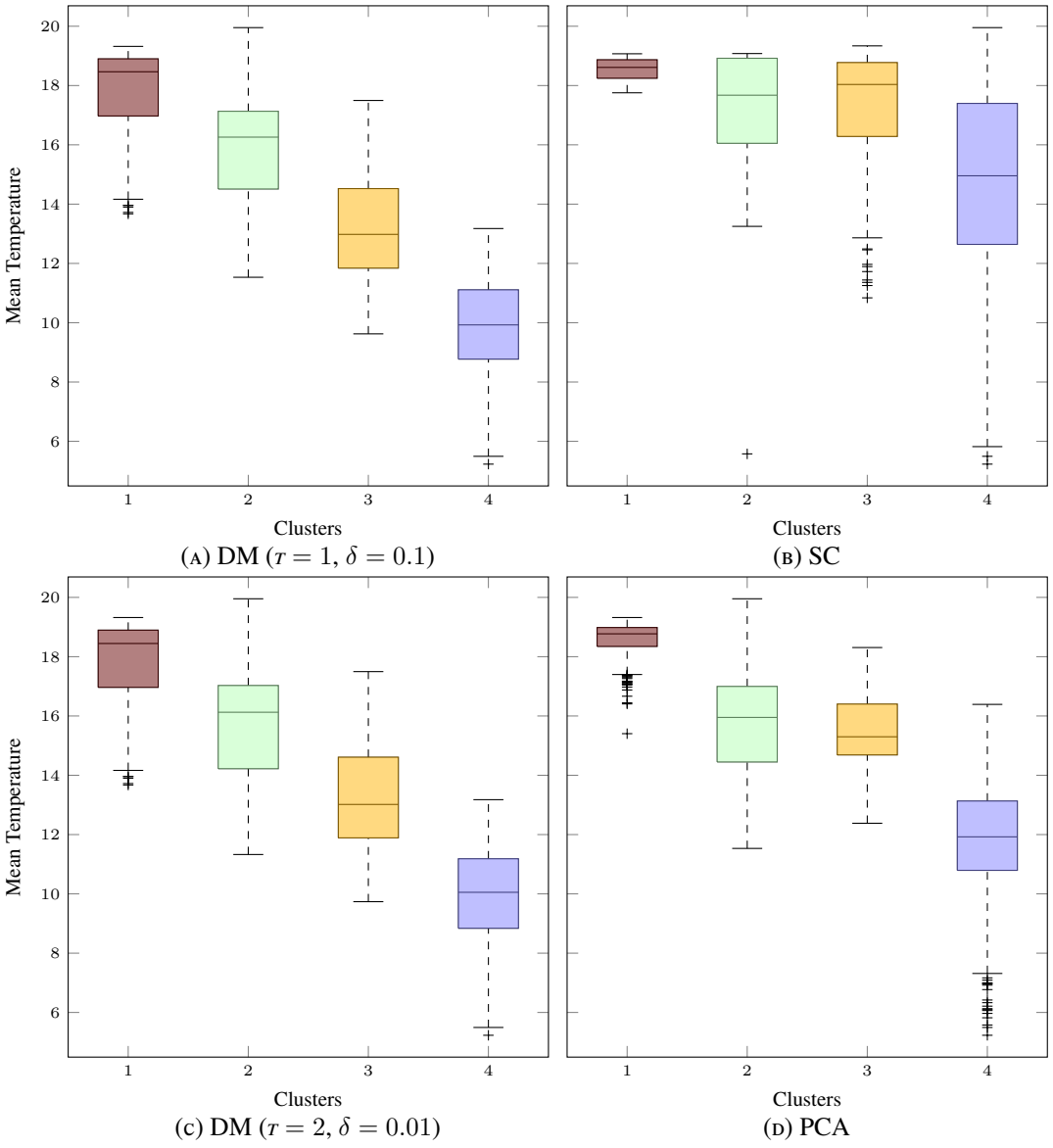
(A) DM ($r = 1$, $\delta = 0.1$) (B) SC



(C) DM ($r = 2$, $\delta = 0.01$) (D) PCA

FIGURE 3.7: Box plots of the geopotential altitude for time compression.

the classical SC model, there is a big box (numbered as cluster 2) which mixes different geopotentials, and which approximately separates the North and South halves of the peninsula. This model does not separate grid points according to the geopotential. Observing the box plots in PCA we have two groups which represent exactly the same geopotential altitude corresponding to the two sea clusters in the map picture, but the other two groups are clearly different and seem to correspond to two different basic land geopotential heights. In the DM models, each cluster appears to be well separated according to the geopotential altitude, and a Mann–Whitney–Wilcoxon test corroborates that each cluster distribution is statistically different. We can thus conclude that DM has indeed been able to greatly reduce the original dimension while capturing meaningful information on the initial patterns.



(A) DM ($r = 1$, $\delta = 0.1$) (B) SC

(C) DM ($r = 2$, $\delta = 0.01$) (D) PCA

FIGURE 3.8: Box plots of the mean temperature for time compression.

It is not easy to interpret what SC and PCA depict as clusters. One possibility is that they concentrate in another variable, but this does not seem to be the case. For instance, we present in Figure 3.8 the temperature box plot diagram for each model. It can be appreciated that neither SC nor PCA are representing temperatures in their clusters, while DM model also capture this feature with their new coordinates. As a possible explanation, we could say then that PCA may be mixing the information and has not separated properly the features that represent this problem.

3.3.3 Spatial Compression

Recall that we are analyzing meteorological data, and in this context, we can consider two different situations depending on where we put an emphasis: a time compression point of view that we have just explained, or a spatial compression of weather forecasts.

Turning to the spatial component of weather forecasts, we will consider for a given day the vector made up of the 8 daily forecasts of the 5 surface variables at each node (recall that the used grid has $1,995$ points). Thus, now we have a sample of 365 points with dimension $79,800 = 1,995 \times 5 \times 8$.

Similarly to the time compression case, when dealing with daily forecasts over a large grid, we should expect that points close in the reduced space will be also close in time, i.e., the reduced representation should somehow reflect, for instance, seasonal features. We will confirm this hypothesis with the following experiment.

3.3.3.1 Parameter Setting

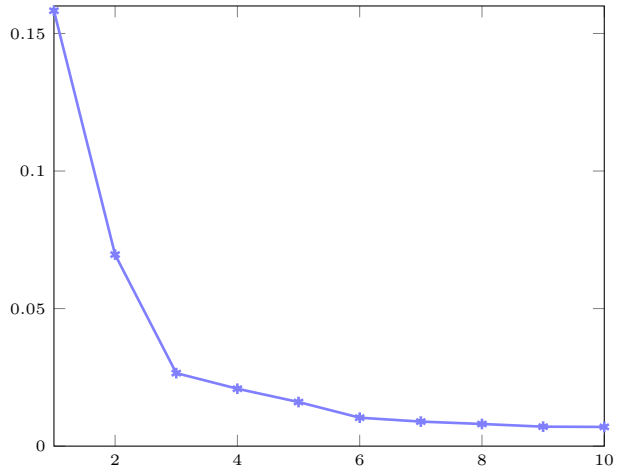
We proceed as before, obtaining now a σ value of 374.40, established again as the median of the distances between all sample points, and determining then the diffusion step τ and its associated dimension $\bar{m}$ at both precisions $\delta = 0.1$ and $\delta = 0.01$.

In Figure 3.9 is shown the eigenvalue decay in the left hand image, and the different reduced dimensions $\bar{m}$ obtained for the different values of each parameter. We also discard in this context a projected dimension of 1, and we settle for a dimension of 5 that we achieve when $\tau = 1$ at the 0.1 precision and when $\tau = 2$ at the 0.01 one. This dimension is in accordance with the eigenvalue decay, where we can appreciate that after the fifth eigenvalue the curve is almost flat and very close to 0. The α parameter has been fixed to 1 for all the models.

We will also use clustering to ascertain the relevance of the reduced dimensions. Since we associate here patterns with calendar days, a logical assumption would be that the reduced patterns should capture seasonal weather behavior. We will apply again κ-means with $\kappa = 4$, although in this case is not only for visualization but also because we are looking for four seasons.

3.3.3.2 Experimental Results

Once the parameters are decided, we apply the different methods to this new context. Recall that we are comparing four models: the classical SC, DM with $\tau = 1$ and $\tau = 2$ and PCA. In this case we will apply PCA with a five-dimensional projection. These five coordinates explain the 45.27% of the variance.



τ	$\delta = 0.1$	$\delta = 0.01$
1	5	72
2	2	5
3	1	2
4	1	2

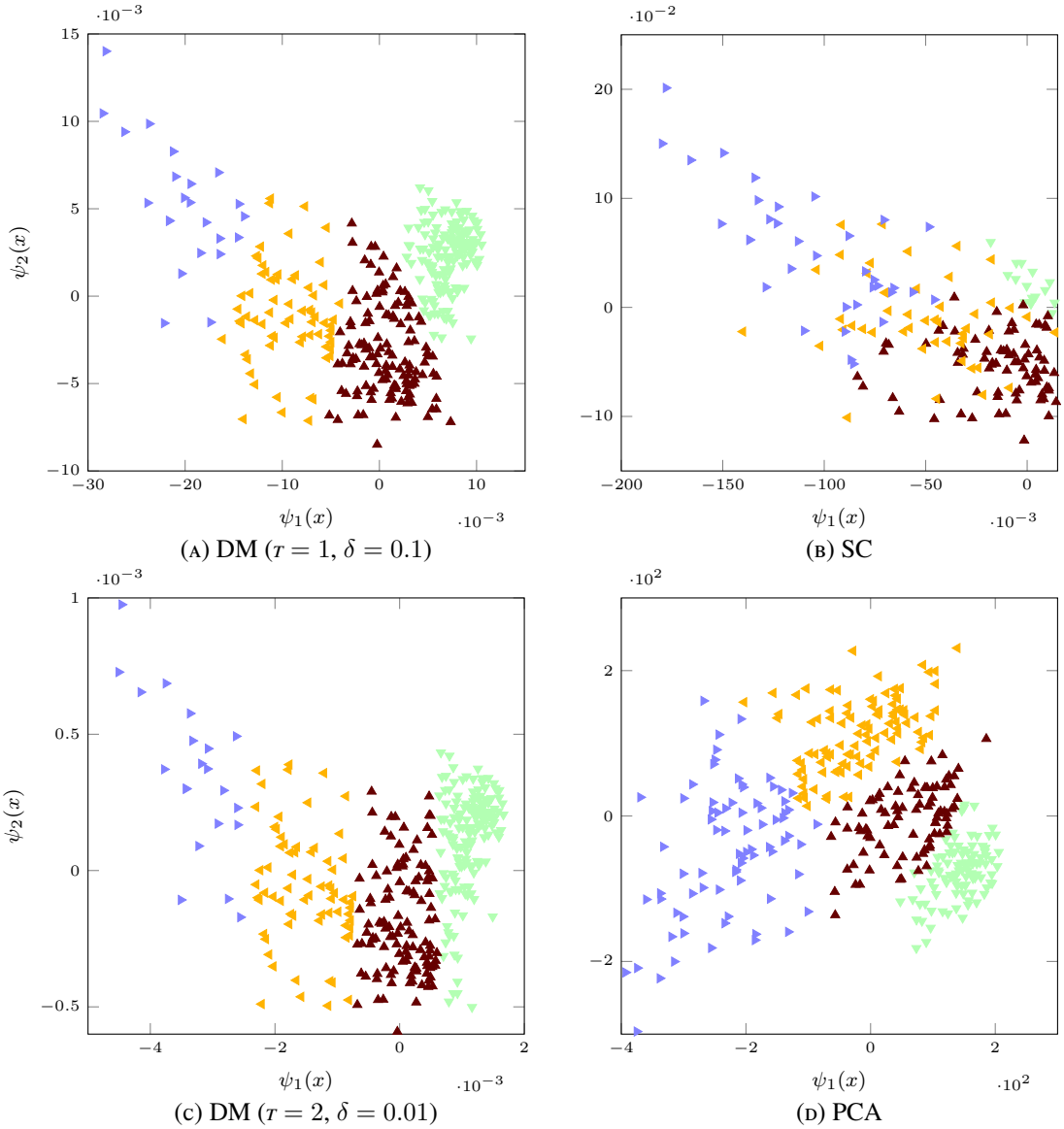
FIGURE 3.9: Parameter setting for spatial compression. The left hand image presents the first 10 eigenvalues (without λ_0) of the probability matrix. The right hand image shows the reduced dimension for some different δ and τ values.

The image in Figure 3.10 shows the two-dimensional representation of the embedded data corresponding to the two largest eigenvalues. The embeddings for the DM models are equal up to the scaling, and they are very similar to the PCA projection, being difficult to decide which of them captures better the structure of the data. All methods but SC yield well defined, relatively compact clusters; SC somehow mixes the clusters in this two-dimensional representation and it seems to make three or higher dimensional associations.

To evaluate properly the embedded coordinates we would like to visualize in some way the impact of these clusters in the original data. Figure 3.11 shows the correspondence between clusters and the seasonal distribution of days. We depict for each "day type" cluster the histogram of the distribution of spring, summer, autumn and winter real days, with the slight modification of considering March, April and May as spring, June, July and August as summer, September, October and November as autumn and December, January and February as winter.

For all the images, the first cluster can be clearly associated with summer, while winter could be associated with the fourth cluster but not in a so clear way. With a slight abuse of language, we will refer to them as the summer and winter clusters. On the other hand, and perhaps not too surprisingly, there is no such clear-cut association of the other two clusters with either spring or autumn. Moreover, notice that the size of the clusters associated to summer is bigger than that of the winter clusters, pointing to the fact that for the Iberian Peninsula, weather summer is longer than calendar summer and the other way around for winter.

In particular, PCA performs better now than for time compression and its model seems to be more balanced than the ones obtained with DM or SC. Looking to its histogram, the four seasons are better and clearer separated, and particularly spring and autumn are better determined than the DM intermediate clusters.



(A) DM ($\tau = 1$, $\delta = 0.1$)

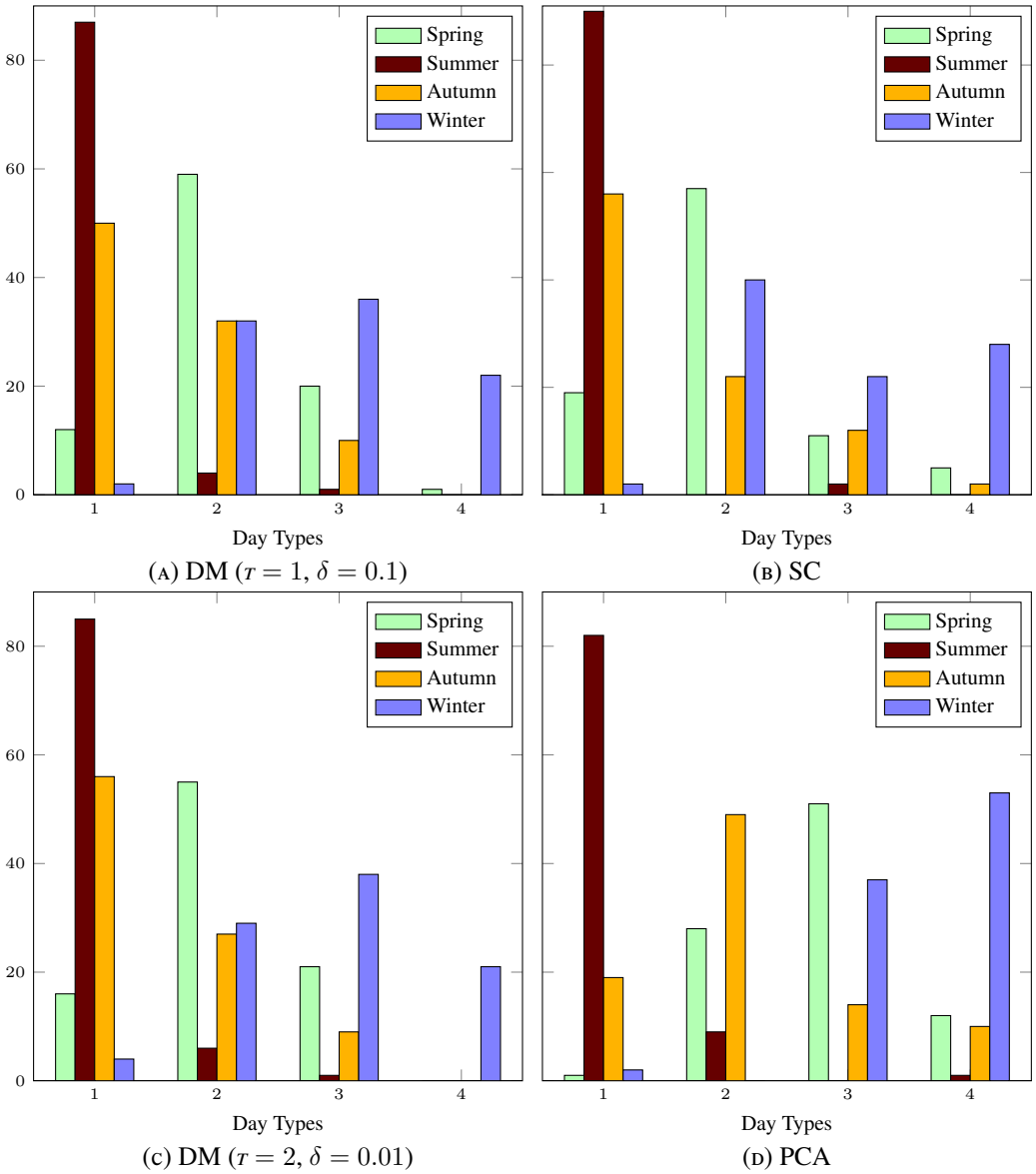
(B) SC

(C) DM ($\tau = 2$, $\delta = 0.01$)

(D) PCA

FIGURE 3.10: Embeddings resulting for spatial compression. The images represent the first two embedded coordinates for four different models.

We also analyze here whether the embedded representations just capture a seasonal variable, most likely temperature, instead of performing a more comprehensive dimensionality reduction. We have considered the daily mean temperature distribution for each one of the four clusters and depicted it in some box plot diagrams, shown in Figure 3.12. DM models box plots corroborate the histogram intuition showing the cluster mean temperatures to be quite close in three cases. They present a logical structure in the sense that the summer cluster has a higher mean temperature than the intermediate clusters, and the winter cluster seems to reflect lower mean temperatures but it has a large overlap with the third cluster's temperatures. A similar behavior could be appreciated in the SC box plots. Nevertheless, the PCA box plot of mean temperatures present a good separation across clusters in a clear seasonal way, with some overlapping points



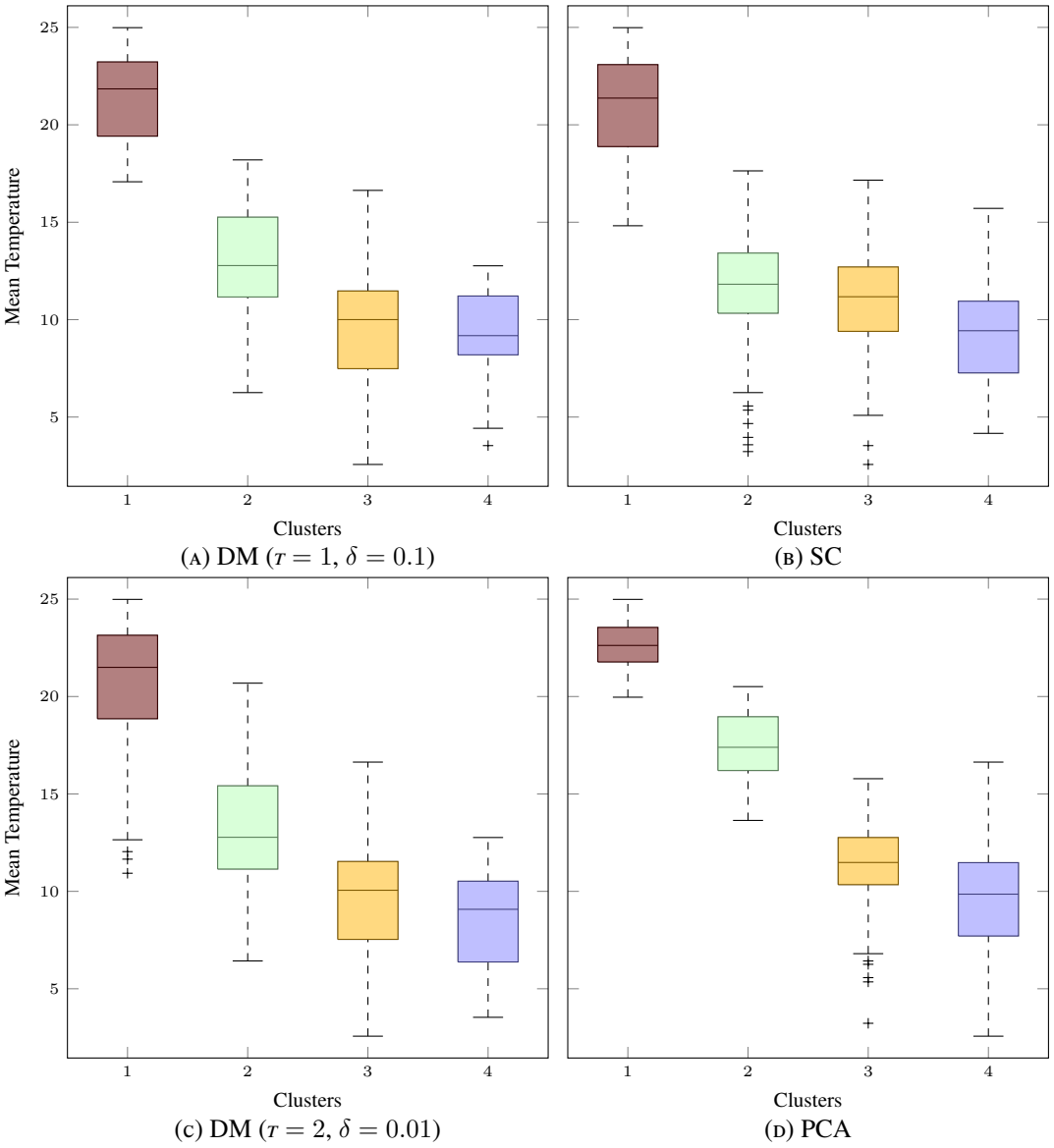
(A) DM ($\tau = 1$, $\delta = 0.1$) (B) SC

(C) DM ($\tau = 2$, $\delta = 0.01$) (D) PCA

FIGURE 3.11: Histograms resulting for spatial compression.

forming a proper transition period.

After the preceding discussion we can conclude that for this problem DM with $\tau = 2$ captures well the underlying information in a very low dimensional description although here PCA does a better job.



(A) DM ($\tau = 1$, $\delta = 0.1$) (B) SC

(C) DM ($\tau = 2$, $\delta = 0.01$) (D) PCA

FIGURE 3.12: Box plots of the mean temperatures for spatial compression.

3.4 DM for Clustering and Local Modeling Wind Power Data

3.4.1 Problem Definition

Local models are a natural and attractive option when trying to approach processes with high variance data or whose underlying phenomena are known to possibly correspond to quite different settings. However, to identify the appropriate local feature areas may be quite difficult, particularly for high dimensional data that do not lend themselves easily to such a task. Unsupervised clustering methods appear as an attractive option. However, clustering is often more an art than a technology and while many methods have been proposed, simple approaches are usually followed in practice. In particular this is the case of κ-means which is applied assuming

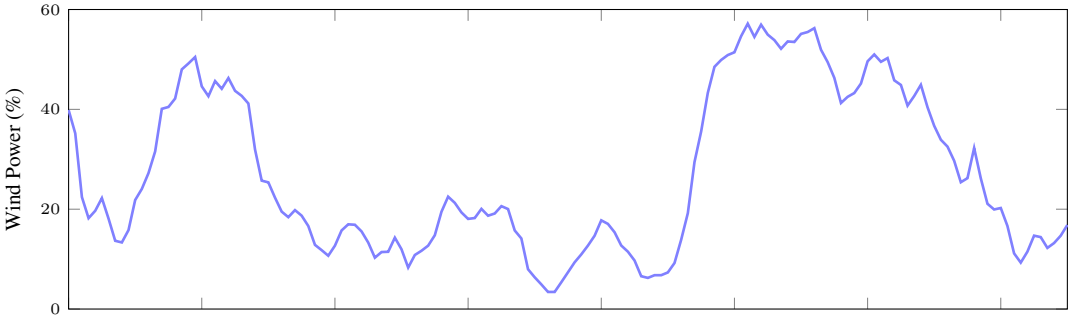


FIGURE 3.13: An example of wind power behavior in Spain.

a Euclidean distance in the feature space. Besides fixing the number κ of clusters, an adequate sampling is also an important issue when working with high dimensional data as samples are then bound to be very sparse. Moreover, the features to be used may not be homogeneous, something probably better to be handled outside the chosen clustering procedure.

These are exactly the problems found when working with wind energy prediction. Wind power clearly presents wide, fast changing fluctuations, certainly at the individual farm level but also when the production of much larger areas is considered (an example can be seen in Figure 3.13). This is the case of Spain, currently one of the world's biggest producers of wind power. The well known, sigmoid-like structure of wind turbine power curves clearly shows different regimes at low, medium, and high wind speeds.

Compounded with this are wind speed frequencies that approximately follow a Weibull distribution, that is, a stretched exponential with low wind having large frequencies (see Figure 3.14). While the above does not directly apply when a wide area is considered, different regimes also appear. Wind energy forecasting for large areas also implies high dimensional features as the predictive variables, that are the outputs of Numerical Weather Predictions (NWP) models such as the ECMWF or NOAA Global Forecast System (GFS) [GFS, 2014] ones, are given for large grids that cover the areas under study. Global models may find it difficult to handle these regimes and local models are a natural alternative [Alaíz et al., 2009; Pinson et al., 2009].

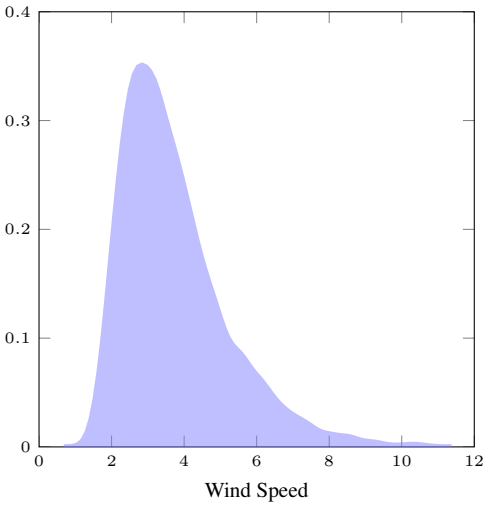


FIGURE 3.14: Weibull distribution of the forecasted wind speed frequencies in the Iberian Peninsula.

This high dimension suggests to precede clustering by some dimensionality reduction technique, preferably, if κ-means is to be used, one that is likely to yield a Euclidean metric for the new features. DM is particularly suited to these requirements. In fact, there is a natural diffusion metric in the original feature space which corresponds with Euclidean metric in the embedded space, as we have seen in Section 3.2. This means that clustering methods that rely on Euclidean metrics, particularly κ-means, should work well on the new features. DM also allows to control to some extent the effects of the underlying data distribution and, moreover, it allows to work with heterogeneous variables. In other words, DM can be a powerful tool for finding informative clusters in high dimensional, heterogeneous data.

Of course, DM is not the only option. Straight κ-means clustering can certainly be used. Moreover, NWP variables for a large area usually show high correlation among different grid points. This may suggest that variance-based dimensionality reduction methods such as PCA may be a useful alternative (we have just seen an example of this in the previous chapter). We shall consider these three options here in order to, first, identify local clusters, and then construct local models to be compared against a global one.

3.4.2 Methodology

3.4.2.1 DM Clusters

The methodology to define local models over the wind energy production will start with a definition of some groups over the data. As explained before, it would be easier to cluster the data if they are appropriately organized. But a good organization is linked to an appropriate distance definition. As the classical κ-means algorithm for clustering behaves well only for Euclidean distances, the data should make sense under this metric. But it is difficult to argue that Euclidean distance is the natural choice for NWP and wind energy against they obeying another Riemannian metric.

These reasons suggest DM as the first step for our experiments, because the diffusion coordinates obtained will maintain the original data structure, but the metric in the new space will be Euclidean. This technique has also the advantage of bringing naturally a dimensionality reduction maintaining the original structure, which allows an easier treatment of the data.

In the Euclidean space defined by the diffusion coordinates we will apply κ-means clustering to determine the regions where the local models will be built using the original coordinates. The result will be compared with the groups obtained by applying κ-means to either the original full dimensional data or to PCA lower dimensional features.

Data. We will use NWP from the ECMWF and wind energy production data p for cluster definition. Wind power is obviously unknown for the test data set so we will use as a proxy the wind power forecast of a global model previously build. We will consider two year data, available over a rectangular 522-point, 0.5° resolution grid that covers the Iberian Peninsula, given 8 times a day. Pattern dimension is thus $2,611 = 5 \times 522 + 1$, as we have 5 different surface variables: wind speed module (V), its horizontal and vertical components (V_x and V_y), pressure (P) and temperature (T), plus the globally predicted wind power.

As the meteorological data available is so heterogeneous, we have taken advantage of the natural way in which homogeneity fits into the diffusion coordinates (see Section 3.2.4). For this, we will consider wind power production and the NWP variables as entries for the DM model as heterogeneous, applying first DM to each feature (p, V, V_x, V_y, T and P) separately and then mix them all to obtain the final features.

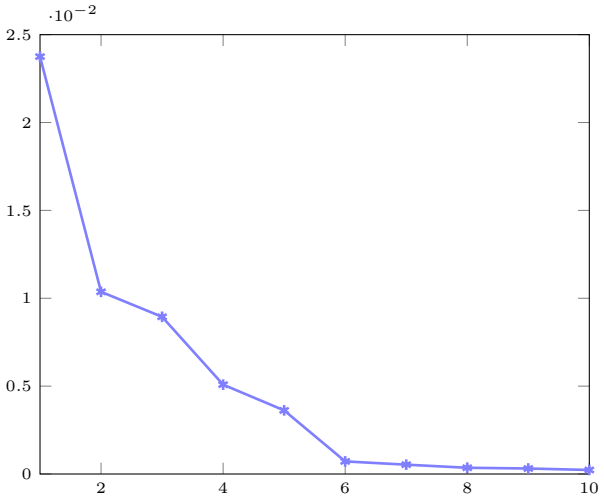
We will skip here the problem of out-of-sample data by building the DM features and clusters, as well as the plain and PCA clusters, over the full two year data set. This confers some advantage to the local models over the global one, partially compensated by the influence that the global model has over the clusters definition.

Parameter setting. For each DM variable-model we have selected a Gaussian kernel to define the graph's similarity matrix, with bandwidth σ equal to the data set diameter. We arrived at this σ value heuristically, after visually analyzing the structure of the resulting embeddings and checking that a bigger value for σ, i.e. considering a bigger neighborhood, visually yields better embedding results. In these models we worked with $\tau = 1$, i.e. considering the one-step diffusion distance on the original feature space, the α parameter is, as always, fixed to 1, and the final embedding dimension was obtained using a $\delta = 0.1$ precision parameter.

In Figure 3.15 the embedding dimensions for each variable and for the final coordinates are shown in the right hand table. The image in the left shows the eigenvalues that correspond to the aggregation of the intermediate diffusion coordinates of each feature. In this eigenvalue curve, it can be seen that a final five-dimensional space is reasonable, as there exist a big decay since the sixth eigenvalue, and the curve tends to zero.

Therefore, for comparison purposes we also considered a $2,611$ to 5 dimension reduction for PCA, which corresponds with a 89.55% of variance explained.

To finish the clustering part of the algorithm, the selection of κ should be made, which is always a difficult task. We have considered in this case 3 clusters that hopefully capture high, medium and low ranges of wind power. While initial centroids are randomly chosen in κ-means, we found that the DM parameters used lead to very stable cluster structures that are essentially independent of centroid initialization.



Variable	$\bar{M}$
V	4
V_x	3
V_y	2
T	1
P	1
p	1
Global	5

FIGURE 3.15: Parameter setting for the DM models over the heterogeneous set of meteorological features. The left hand image presents the first 10 eigenvalues (without λ_0) of the similarity matrix for the aggregation of the selected coordinates over each feature. The right hand image shows the reduced dimension for each feature and for the final aggregation, using a value of $\tau = 1$ and $\delta = 0.1$.

3.4.2.2 Local models

Once the data is clustered, models should be defined. Many paradigms can be considered for model building but we will concentrate first on the simplest alternative, Ridge Regression (RR), also known as Tikhonov Regularized Least Squares (RLS) [Hoerl and Kennard, 1970], certainly not the strongest possible method but a good option to measure the usefulness of local methods against a global one.

The chosen model adds a ℓ_2 regularization term to an Ordinary Linear Least Squares (OLS) regression, so the optimization problem becomes

$$\min_{\mathbf{X}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \gamma \|\mathbf{w}\|_2^2, \tag{3.10}$$

whose explicit solution is given by

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X} + \gamma \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}.$$

These models are homogeneous, in the sense that they do not consider any bias term. Actually, this term can be always omitted just by centering the features and the targets around zero. However, sometimes is useful to include this term, as in this experiment, where we are interested on respecting the natural scaling of the features. It is easy to introduced a bias term in the model

explained, adding to the matrix input a constant 1 entry,

$$\mathbf{X}_b = \begin{pmatrix} \mathbf{1} & \mathbf{X} \end{pmatrix},$$

where $\mathbf{1}$ is a column vector equal to one. The solution in this case will be given by

$$\begin{pmatrix} b^* \\ \mathbf{w}^* \end{pmatrix} = \left(\mathbf{X}_b^\top \mathbf{X}_b + \frac{\gamma}{\bar{M}} \begin{pmatrix} 0 & 0 \\ 0 & I \end{pmatrix} \right)^{-1} \mathbf{X}_b^\top \mathbf{y}.$$

The ℓ_2 term penalizes the complexity of the model, avoiding possible overfitting of standard OLS. The Tikhonov parameter γ determines the trade-off between regularization and the bias of the model. When $\gamma = 0$, we recover the classical (unregularized) OLS formulation. On the other side, when γ grows, $W^* \to 0$ and the model will eventually become constant.

We will compare a global RR model and also local RR models built over the different clusters. We will denote them as $\mathcal{GM}$ for the Global model, $\mathcal{LM}_{\text{DM}}$ for the DM clusters, $\mathcal{LM}_{\text{PC}}$ for the PCA clusters and $\mathcal{LM}_{\text{Or}}$ for the models constructed over the clusters defined in the original features.

As usually done in wind energy, the model performance is measured by the Mean Absolute Error (MAE) and the Relative Mean Absolute Error (RMAE). The MAE is defined as the mean of the absolute values of the differences between the predictions $\hat{y}_i$ and the real values y_i:

$$\mathcal{E}_{\text{mae}} = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{N}.$$

The RMAE computes the same mean for the ratio of the absolute errors over the actual wind power:

$$\mathcal{E}_{\text{rmae}} = \frac{\sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{y_i}}{N}.$$

Data. To build local models we use as inputs the NWP from the ECMWF, considering among them five surface variables: wind speed module (V), its horizontal and vertical components (V_x and V_y), pressure (P) and temperature (T). These variables are available over a rectangular 522-point, 0.5° resolution grid that covers the Iberian Peninsula. Pattern dimension is thus $2,610 = 5 \times 522$. Two-year data will be considered, the first one for training purposes and the second one for testing. Since eight forecasts are given daily, training sample size is thus $2,910 = 365 \times 8$, close to feature dimension and hence making regularization mandatory.

All the model wills be built over variables normalized component-wise to zero mean and unit variance.

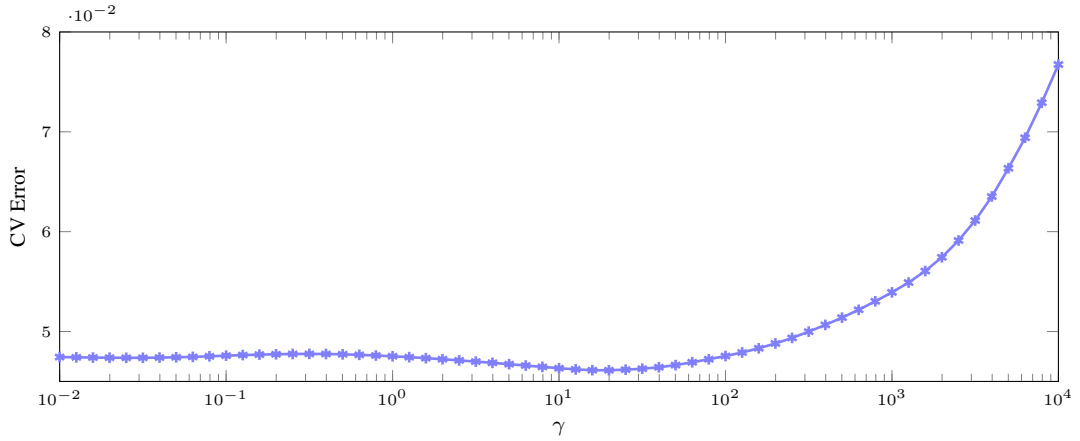


FIGURE 3.16: Cross validation error for the parameter γ search for the global model.

Parameter Setting. To build the above models over the data available, we have to select the adequate parameters. For model building the optimal regularization parameters for all the RR models should also be selected. This step has been made by a grid search for γ in the interval $[10^{-2}, 10^{4}]$, with a logarithmic step of 0.1 and using as validation set the last 20% patterns of the first year data clusters.

In Figure 3.16 we show the error curve obtained during the cross validation step for the global model. The optimal parameter has been fixed in $\gamma = 19.95$, which makes sense in terms of the error curve. This has been repeated for each local model, obtaining in the same way an appropriate γ value.

3.4.3 Experimental Results

The algorithm proposed has been run with the parameters selected. The result of the first step, the final Diffusion embedding where we shall work and the clusters defined over it, is shown in Figure 3.17.

It can be seen that the embedded space captures the original data in a structured and compact way, with some outliers probably of points belonging to the high wind region. The clusters defined are difficult to evaluate over this space, so we try to represent the results over the original data. Figure 3.18 gives the empirical density function of the local wind module and wind power over the clusters for each approach. As it can be seen, the three DM clusters offer a more clear-cut structure for both features while the other two methods seem to make no clear difference between regimes of any type, as the clusters present a higher degree of overlapping. Thus, it seems clear that the DM correspond to the low, medium and high wind and energy regimes.

In these figures, it can be also observed that the separation in wind energy regions is more clear than the separation over wind module regions. This is coherent with the results obtained in

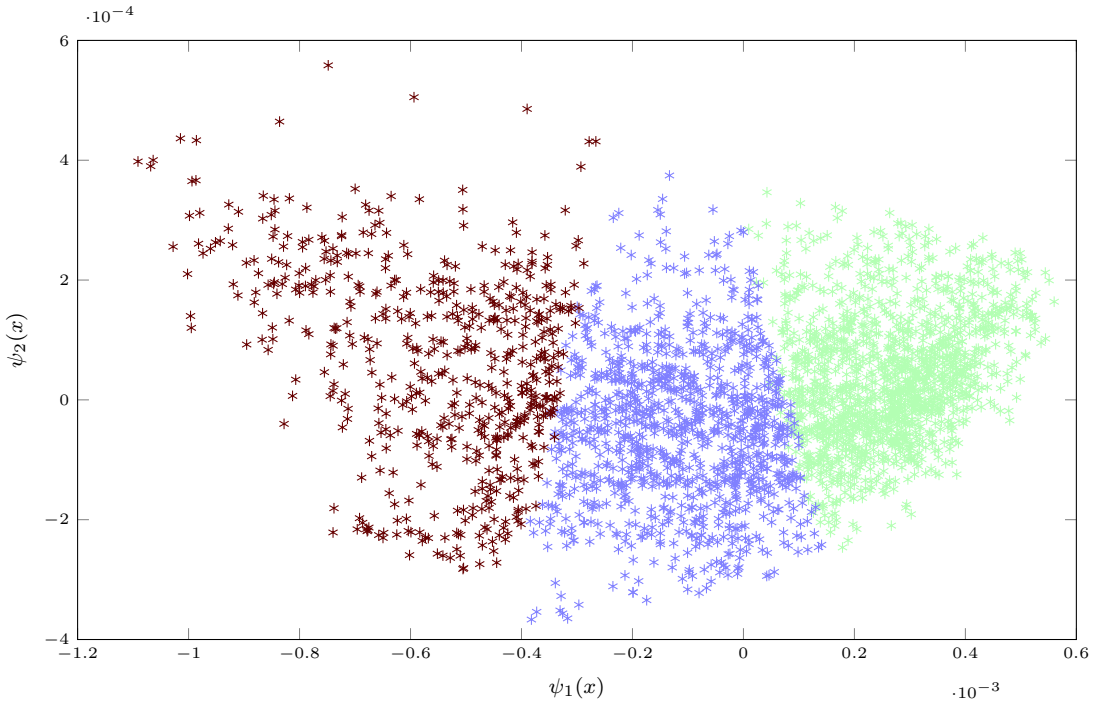


FIGURE 3.17: DM embedding for local model definition.

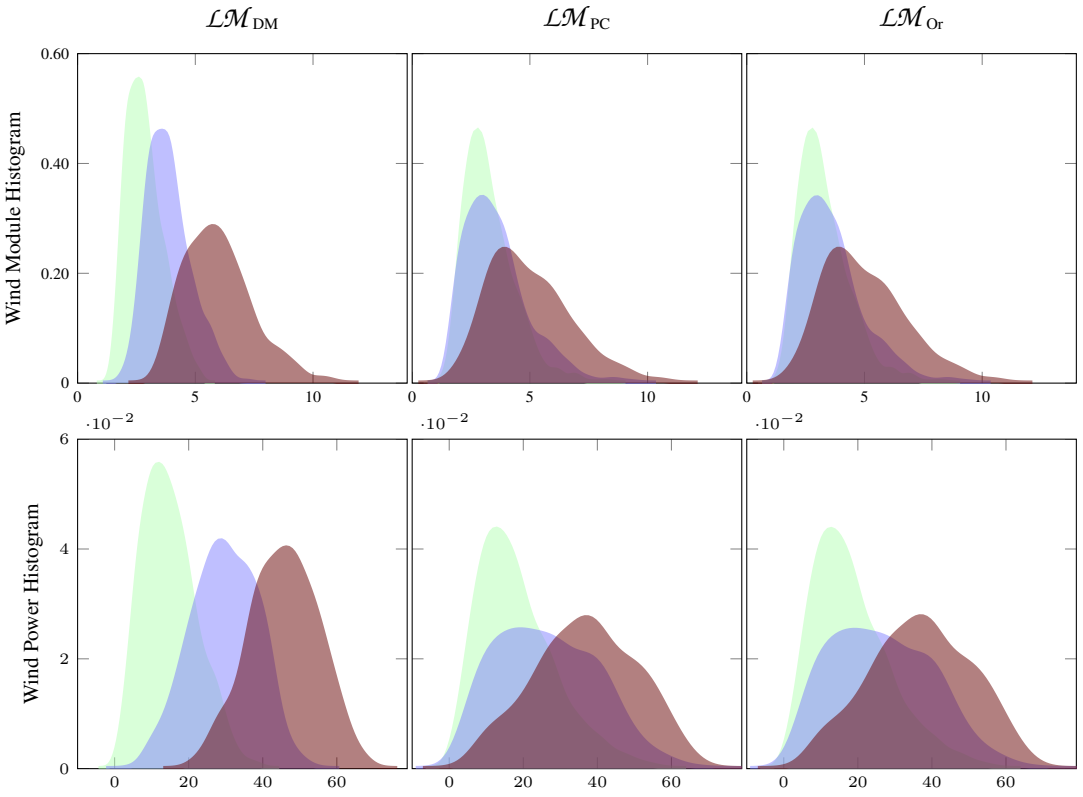


FIGURE 3.18: Histograms for the Wind Module (top) and for the Wind Power (bottom).

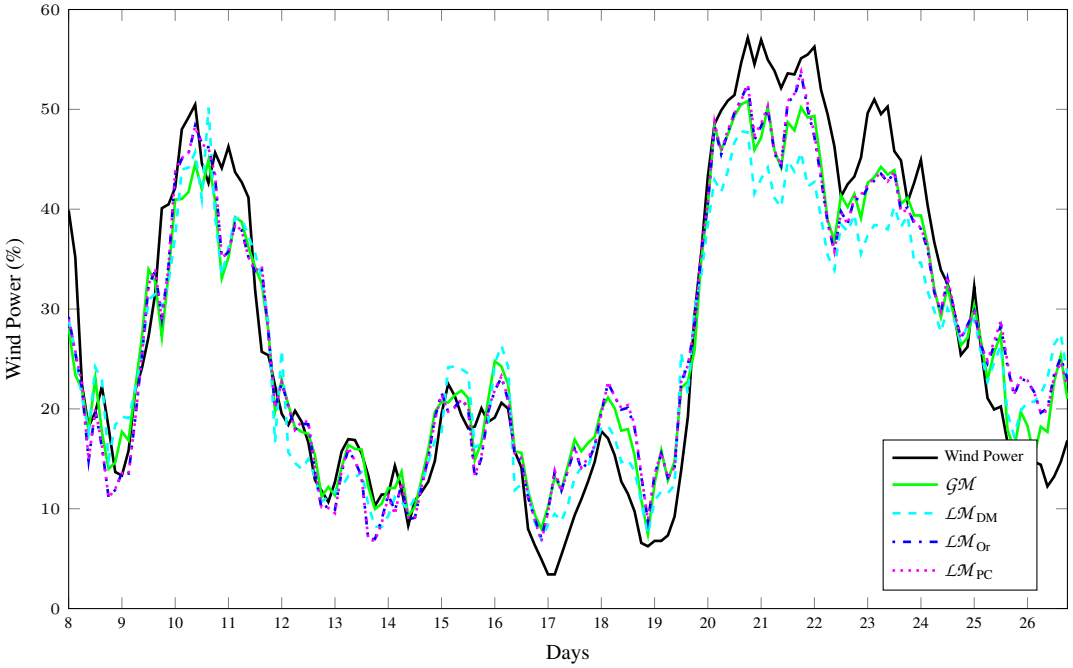


FIGURE 3.19: Comparative curves of the different models employed for some September days.

[Barbero et al., 2008] where it has been shown that wind module regimes not necessarily define wind power regimes, which are the ones we are interested on.

An example of the behavior in testing of the different models for some days of September can be seen in Figure 3.19. It can be appreciated that all models follow the tendency of the real energy curve and it can be seen how in this period all the models shown a similar behavior.

Table 3.2 contains local model errors per cluster as well as the cluster errors of the global model. The clusters have been ordered for every model, in such a way that cluster $\mathscr{C}_1$ corresponds with the low wind power cluster, $\mathscr{C}_2$ represents the medium regime and $\mathscr{C}_3$ contains the higher range of wind power. Table 3.2 also gives the standard deviations of MAE and RMAE, although they are rather conservative (assuming independence for these errors would lead to divide the values given by the square root of sample size and, hence, much smaller values).

As we can see, the local models beat the global one in the first, low wind power cluster $\mathscr{C}_1$. $\mathcal{LM}_{\mathrm{DM}}$ also beats the global model in the high wind power region $\mathscr{C}_3$, while in the medium one, $\mathcal{GM}$ is clearly better. This effect is probably due to the more specific characteristics of the lowest and highest wind powers, i.e., looking at the smoothed histogram distributions for the $\mathcal{LM}_{\mathrm{DM}}$ model, we can appreciate that these two clusters correspond to points with more localized wind power values than the medium cluster, whose productions are more mixed with those of the other clusters.

TABLE 3.2: Local and global wind model errors per cluster.

	MAE		RMAE	
	$\mathcal{LM}_{\text{DM}}$	$\mathcal{GM}$	$\mathcal{LM}_{\text{DM}}$	$\mathcal{GM}$
$\mathscr{C}_1$	$\mathbf{2.44 \pm 1.99}$	2.75 ± 2.21	$\mathbf{21.09 \pm 32.73}$	25.42 ± 42.99
$\mathscr{C}_2$	5.09 ± 3.85	$\mathbf{4.43 \pm 3.59}$	25.22 ± 80.18	$\mathbf{20.59 \pm 67.26}$
$\mathscr{C}_3$	$\mathbf{5.79 \pm 4.95}$	5.94 ± 5.17	15.13 ± 23.45	$\mathbf{14.17 \pm 21.77}$
	$\mathcal{LM}_{\text{Or}}$	$\mathcal{GM}$	$\mathcal{LM}_{\text{Or}}$	$\mathcal{GM}$
$\mathscr{C}_1$	$\mathbf{2.52 \pm 2.13}$	2.77 ± 2.36	$\mathbf{18.87 \pm 23.39}$	20.48 ± 23.91
$\mathscr{C}_2$	$\mathbf{3.82 \pm 3.08}$	3.94 ± 3.24	$\mathbf{20.47 \pm 37.53}$	21.45 ± 41.46
$\mathscr{C}_3$	6.59 ± 4.80	$\mathbf{5.44 \pm 4.68}$	36.95 ± 120.23	$\mathbf{26.87 \pm 97.23}$
	$\mathcal{LM}_{\text{PC}}$	$\mathcal{GM}$	$\mathcal{LM}_{\text{PC}}$	$\mathcal{GM}$
$\mathscr{C}_1$	$\mathbf{2.52 \pm 2.13}$	2.78 ± 2.36	$\mathbf{18.91 \pm 23.36}$	20.55 ± 23.94
$\mathscr{C}_2$	$\mathbf{3.83 \pm 3.08}$	3.94 ± 3.25	$\mathbf{20.53 \pm 37.69}$	21.43 ± 41.52
$\mathscr{C}_3$	6.48 ± 4.81	$\mathbf{5.40 \pm 4.67}$	35.98 ± 116.73	$\mathbf{26.70 \pm 96.58}$

In Figure 3.20 we depict the relationships between average wind and power (top) and between average wind and predicted power (bottom) for the three $\mathcal{LM}_{\text{DM}}$ clusters. Cluster $\mathscr{C}_3$ presents several marked outliers, which are thus more difficult to predict. That is the reason why the errors in this cluster are much higher. It can be also seen that cluster $\mathscr{C}_2$ is broader and less specific, making it more difficult to forecast with a local model, as the global model may have more information of this intermediate state.

The $\mathcal{LM}_{\text{PC}}$ and $\mathcal{LM}_{\text{Or}}$ models outperform the results obtained by $\mathcal{GM}$ in clusters $\mathscr{C}_1$ and $\mathscr{C}_2$, but lose in the high wind power region. If we observe Figure 3.18, we can appreciate that these models do not separate the first two clusters; moreover, their third cluster is rather small. Thus, in the first two clusters they may be just building global models without the outliers, which correspond to the highest wind power values, outperforming then the global model that has to handle these outliers. But since their third cluster is small, they seem to overfit it and, thus, lose there against the more balanced global model. Considering these results, it can be said that $\mathcal{LM}_{\text{DM}}$ gives more consistent clusters that clearly divide wind module and wind power regimes.

All these facts suggest building predictors combining local models and the global one according to their performance in each cluster. Table 3.3 contains the MAE and RMAE errors of the individual $\mathcal{GM}$, $\mathcal{LM}_{\text{DM}}$, $\mathcal{LM}_{\text{Or}}$ and $\mathcal{LM}_{\text{PC}}$, and of the combined models $\mathcal{CM}_{\mathcal{LM}_{\text{DM}};\mathcal{GM}}$, $\mathcal{CM}_{\mathcal{LM}_{\text{Or}};\mathcal{GM}}$ and $\mathcal{CM}_{\mathcal{LM}_{\text{PC}};\mathcal{GM}}$.

It shows that there is a clear advantage of the combined models over the global ones and that the gain is largest for the $\mathcal{CM}_{\mathcal{LM}_{\text{DM}};\mathcal{GM}}$ model. While modest at first sight (a MAE of 3.64% against 3.81% for $\mathcal{GM}$), such gains may have a large economic impact as we are considering the prediction over all Spain.

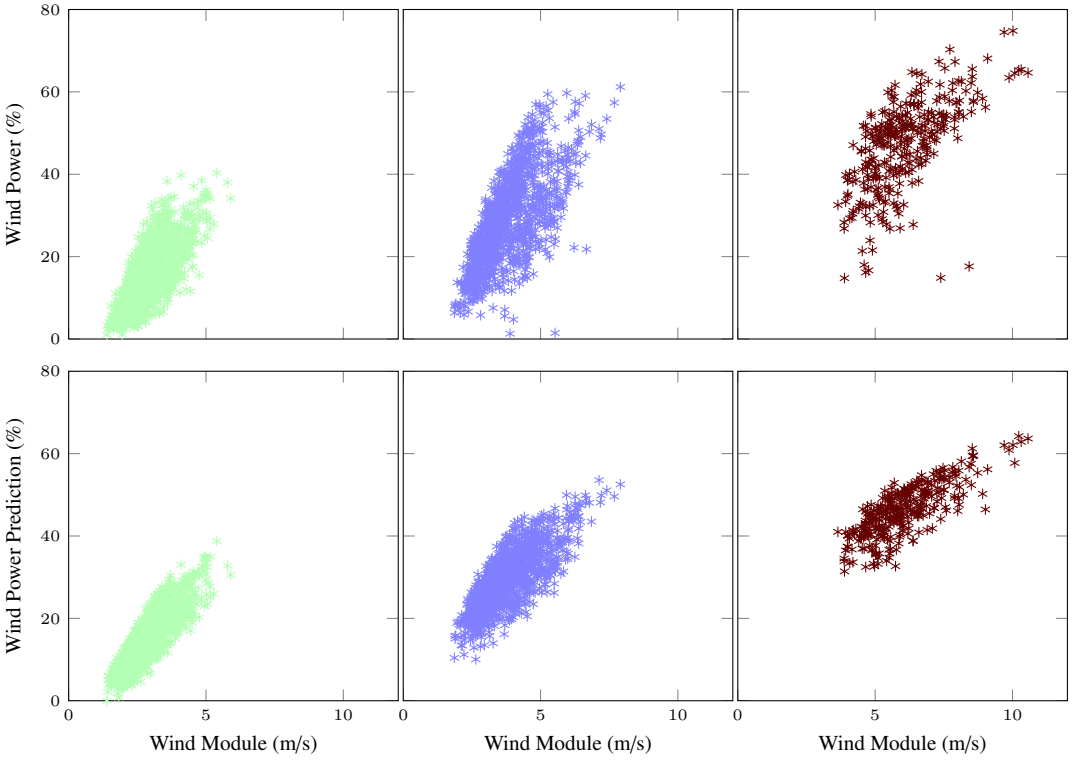


FIGURE 3.20: Wind Power Real Curve (top) versus Wind Power Predicted Curve (bottom) for the three $\mathcal{LM}_{\text{DM}}$ clusters.

TABLE 3.3: Global errors for the local, global and combined wind models.

	$\mathcal{GM}$	$\mathcal{LM}_{\text{DM}}$	$\mathcal{LM}_{\text{Or}}$	$\mathcal{LM}_{\text{PC}}$	$\mathcal{CM}_{\mathcal{LM}_{\text{DM}};\mathcal{GM}}$	$\mathcal{CM}_{\mathcal{LM}_{\text{Or}};\mathcal{GM}}$	$\mathcal{CM}_{\mathcal{LM}_{\text{PC}};\mathcal{GM}}$
MAE	3.81 ± 3.12	3.92 ± 3.09	3.89 ± 3.07	3.88 ± 3.08	$\mathbf{3.64 \pm 2.99}$	3.66 ± 3.05	3.66 ± 3.05
RMAE	22.15 ± 50.67	22.13 ± 51.38	23.11 ± 48.54	23.01 ± 48.15	$\mathbf{20.21 \pm 46.01}$	21.14 ± 44.04	21.16 ± 44.15

In summary, we can conclude that DM dimensionality reduction and clustering is an effective tool for local model building in wind energy forecasting.

Chapter 4

Anisotropic Diffusion

4.1 Introduction

In general, diffusion methods assume that sample values lie in a manifold $\mathscr{M}$ whose geometry corresponds to a diffusion distance associated with a Markov process. The Diffusion Maps (DM) model presented in Chapter 3 is an example of this kind of methods, but it is not the only one. In this chapter we will present Anisotropic Diffusion (AD), a technique that assumes a different underlying model where there exist some latent variables connected to the observable ones by an unknown function.

The principal difference between both methods resides in their basic assumption. In the isotropic case (the DM method) it is assumed that the data lie on a Riemannian manifold whose metric is equivalent to a certain diffusion distance over the ambient feature space. The first step in its application is thus to define a Markov probability matrix starting from an initial similarity given in terms of the heat kernel. The eigenanalysis of the Markov matrix enables to define a new set of diffusion coordinates for which the Euclidean distance coincides with the diffusion distance in the sample space. In other words, the diffusion map transforms the natural diffusion metric on the sample manifold into Euclidean distance on the diffusion coordinates.

On the other hand, AD assumes that the sample is generated by a nonlinear function f acting on some latent features that follow an Itô process. These latent parameters can be obtained inverting f in a kind of nonlinear component analysis, something that is achieved starting from

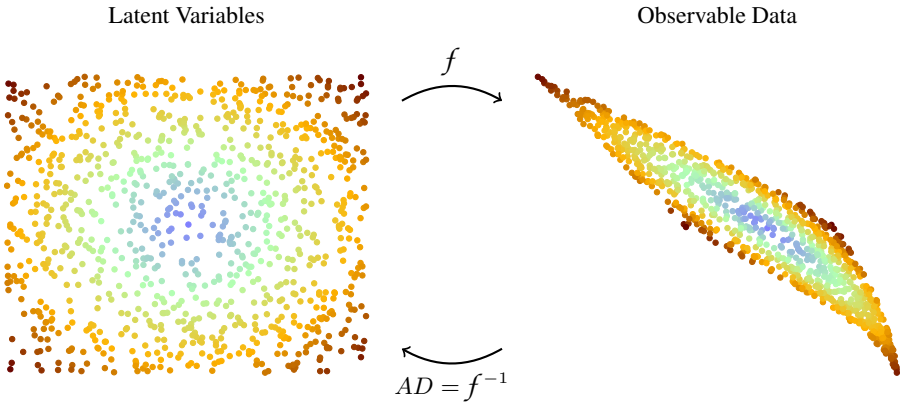


FIGURE 4.1: AD operation example.

a similarity matrix and its corresponding sample graph in which Euclidean distance in the standard heat kernel is replaced by Mahalanobis distance with respect to a local sample covariance matrix $\mathbf{C}$. More precisely, and as in isotropic diffusion, the original parametric features can be recovered by an appropriate eigenanalysis of the anisotropic Markov matrix that is related in this case to the Fokker–Planck operator associated to the Laplacian manifold.

An example of how AD works is shown in Figure 4.1. In this case, the unknown, latent variables that we would like to recover are represented on the left hand image and, on the right hand, the observable data is depicted, which is the result of a nonlinear combination of the latent variables. In this concrete example, taken out from Singer and Coifman [2008], the latent variables are some random points uniformly distributed and the function f that generates the observable data is defined by

$$f_1 = \sin(2x_1 - x_2)$$
$$f_2 = \sin(x_2 - x_1).$$

AD will try to recover the inverse f^{-1}, such that we come back to the original, independent coordinates (or to a equivalent representation).

However, in both diffusion methods the recovery of the intrinsic features requires a computationally very costly eigenanalysis of the Markov transition matrix. On the other hand, this recovery is not strictly needed in some problems and we will pursue here an alternative way of applying AD for them, building models that are based on a κ-Nearest Neighbors (κ-NN) search defined in terms of the anisotropic diffusion metric. This allows to estimate the Euclidean distance in the inaccessible latent space through local Mahalanobis distances in the sample manifold $\mathscr{M}$ without having to go through any costly eigenanalysis, mandatory in the classical diffusion methods.

4.2 Theoretical Background

In Anisotropic Diffusion (AD) we assume that the observable patterns $\mathscr{S} = \{x^{(1)}, \ldots, x^{(N)}\} \subset \mathscr{M}$ are the result of a nonlinear, smooth and bi-Lipschitz transformation $f : \mathscr{L} \to \mathscr{M}$ of patterns $\boldsymbol{\ell}^{(p)} \in \mathscr{L}$ that correspond to latent features. Under this model, the main goal is to define a new distance in the observable space that approximates the Euclidean distance between points in the parametric space.

The basic assumption for this theory is that the parametric features $\ell_i,\ i = \{1, \ldots, \bar{M}\}$ follow an Itô process. We will first try to estimate the Euclidean parametric distance in Section 4.2.1, we review in Section 4.2.2 some concepts in Itô calculus to get an approximation over the observable space, and we arrive finally to the AD process.

4.2.1 Estimating the Latent Variable Metric

We are interested in defining an appropriate distance in the observable space equivalent to the Euclidean distance in the parametric space. For this purpose, following Singer and Coifman [2008], let $\boldsymbol{\ell}, \lambda \in \mathscr{L}$ be such that $\mathbf{x} = f(\boldsymbol{\ell})$ and $\xi = f(\lambda)$, and let $g(\mathbf{x}) = f^{-1}(\mathbf{x})$.

We want to estimate $\|\lambda - \boldsymbol{\ell}\|^2 = \|g(\xi) - g(x)\|^2$, which is an easy task applying Taylor series. Expanding $\boldsymbol{\ell} = g(\mathbf{x})$ at ξ, we have for the i-th component

$$\begin{aligned} g_i(\mathbf{x}) = \ell_i = g_i(\xi) - \sum_j \frac{\partial g_i}{\partial x_j}(\xi)(\xi_j - x_j) \\ + \frac{1}{2}\sum_{k,\ell} \frac{\partial^2 g_i}{\partial x_k \partial x_\ell}(\xi)(\xi_k - x_k)(\xi_\ell - x_\ell) + O(\|\xi - \mathbf{x}\|^3). \end{aligned}$$

We arrive now at $\|\lambda - \boldsymbol{\ell}\|^2 = \sum_i (\lambda_i - \ell_i)^2$, which is estimated as

$$\begin{aligned} \|\lambda - \boldsymbol{\ell}\|^2 = \sum_{i,j,k} \frac{\partial g_i}{\partial x_j}(\xi)\frac{\partial g_i}{\partial x_k}(\xi)(\xi_j - x_j)(\xi_k - x_k) \\ - \sum_{i,j,k,\ell} \frac{\partial g_i}{\partial x_j}(\xi)\frac{\partial^2 g_i}{\partial x_k \partial x_\ell}(\xi)(\xi_j - x_j)(\xi_k - x_k)(\xi_\ell - x_\ell) + O(\|\xi - \mathbf{x}\|^4). \end{aligned}$$

In the same way we can expand $\lambda = \mathcal{g}(\xi)$ at $\mathbf{x}$ and we get the alternative approximate expansion

$$\begin{aligned}\mathcal{g}_i(\xi) = \lambda_i &= \mathcal{g}_i(\mathbf{x}) + \sum_{i,j} \frac{\partial \mathcal{g}_i}{\partial x_j}(\mathbf{x})(\xi_j - x_j) \\ &\quad + \frac{1}{2}\sum_{i,k,\ell} \frac{\partial^2 \mathcal{g}_i}{\partial x_k \partial x_\ell}(\mathbf{x})(\xi_k - x_k)(\xi_\ell - x_\ell) + O(\|\xi - \mathbf{x}\|^3) \\ \Rightarrow \|\lambda - \boldsymbol{\ell}\|^2 &= \sum_{i,j,k} \frac{\partial \mathcal{g}_i}{\partial x_j}(\mathbf{x}) \frac{\partial \mathcal{g}_i}{\partial x_k}(\mathbf{x})(\xi_j - x_j)(\xi_k - x_k) \\ &\quad + \sum_{i,j,k,\ell} \frac{\partial \mathcal{g}_i}{\partial x_j}(\mathbf{x}) \frac{\partial^2 \mathcal{g}_i}{\partial x_k \partial x_\ell}(\mathbf{x})(\xi_j - x_j)(\xi_k - x_k)(\xi_\ell - x_\ell) + O(\|\xi - \mathbf{x}\|^4).\end{aligned}$$

Averaging both estimations, we arrive to

$$\begin{aligned}\|\lambda - \boldsymbol{\ell}\|^2 &= \frac{1}{2}(\xi - \mathbf{x})^\top [\mathbf{J}_{\mathcal{g}}^\top \mathbf{J}_{\mathcal{g}}(\mathbf{x}) + \mathbf{J}_{\mathcal{g}}^\top \mathbf{J}_{\mathcal{g}}(\xi)](\xi - \mathbf{x}) + O(\|\xi - \mathbf{x}\|^4) \\ &= \frac{1}{2}(\xi - \mathbf{x})^\top [(\mathbf{J}_f \mathbf{J}_f^\top)^{-1}(\mathbf{x}) + (\mathbf{J}_f \mathbf{J}_f^\top)^{-1}(\xi)](\xi - \mathbf{x}) + O(\|\xi - \mathbf{x}\|^4), \qquad (4.1)\end{aligned}$$

where $\mathbf{J}_{\mathcal{g}}$ and $\mathbf{J}_f$ represent the Jacobian matrices of $\mathcal{g}$ and f, i.e. the distortions of the transformations $\mathcal{g}$ and f respectively. Note that for obtaining Equation (4.1) we have employed that $\mathbf{J}_{\mathcal{g}} = \mathbf{J}_f^{-1}$ and we can include the term involving the Hessian products in the estimate $O(\|\xi - \mathbf{x}\|^4)$, for a first order Taylor expansion yields $\frac{\partial \mathcal{g}_i}{\partial x_j}(\mathbf{x}) \frac{\partial^2 \mathcal{g}_i}{\partial x_k \partial x_\ell}(\mathbf{x}) - \frac{\partial \mathcal{g}_i}{\partial x_j}(\xi) \frac{\partial^2 \mathcal{g}_i}{\partial x_k \partial x_\ell}(\xi) = O(\|\xi - \mathbf{x}\|)$. In what follows we will denote the Jacobian matrix of function f as just $\mathbf{J}$.

The problem is that we need to compute $\mathbf{J}\mathbf{J}^\top$, but the function f is unknown. To compute it in the observable space, we will use our assumption that the latent variables follow an Itô process and also a bit of Itô calculus that we briefly review in the next section.

4.2.2 Itô Processes for Computing Distances over the Observable Space

We start recalling what a Brownian motion is [Rogers and Williams, 2000a, Chapter 1.1].

Definition 4.1 (Brownian motion)**.** A real-valued stochastic process $W(\cdot)$ is called a *Brownian motion* or *Wiener process* if

(i) $W(0) = 0$,

(ii) $W(t) - W(s)$ is $\mathcal{N}(0,\ (t-s)I) \quad \forall t \geq s \geq 0$, i.e., is a multivariate normal with zero mean and $t - s$ variance,

(iii) it has independent increments: for all t_i $0 < t_1 < t_2 < \ldots < t_N$, the random variables $W(t_1), W(t_2) - W(t_1), \ldots, W(t_N) - W(t_N - 1)$ are independent.

Notice in particular that $\mathrm{E}_W(W(t)) = 0$ and $\mathrm{E}_W(W^2(t)) = t$ for each $t \geq 0$.

We need this concept to define Itô processes [Rogers and Williams, 2000b, Chapter 4.1], which are the kind of differential stochastic processes which we are interested in.

Definition 4.2 (Itô process)**.** An *Itô process* is an stochastic process X_t of the form

$$X_t = X_0 + \int_0^t \mathcal{a}(s, X_s)\, ds + \int_0^t \mathcal{b}(s, X_s)\, dW_s, \quad t \geq 0,$$

where W_t is a Brownian motion. It can be rewritten in differential notation as

$$dX_t = \mathcal{a}_t\, dt + \mathcal{b}_t\, dW_t.$$

In this equation, $\mathcal{a}_t$ represents the drift of the process (the deterministic part) and $\mathcal{b}_t$ represents its variance (the stochastic part).

Our assumption for the latent variables is that they follow an Itô process, which means that for each feature ℓ_i we have

$$d\ell_i = \mathcal{a}_i\, dt + \mathcal{b}_i\, dW_i.$$

An important result in Itô calculus that we are interested in is Itô's lemma, which claims that a smooth function of an Itô process is itself an Itô process.

Theorem 4.1 (Itô's Lemma)**.** *Suppose $f(t, X_t)$ is a vector valued twice continuously differentiable function. Then $f(t, X_t)$ is also an Itô process, expressed by*

$$df(t, X_t) = \left(\frac{\partial f}{\partial t} + \mathcal{a}_t \frac{\partial f}{\partial x} + \frac{1}{2} \mathcal{b}_t^2 \frac{\partial^2 f}{\partial x^2} \right) dt + \mathcal{b}_t \frac{\partial f}{\partial x}\, dW_t.$$

Applying Theorem 4.1 and taking into account that in our case f does not depend on t, we can say that the M-dimensional observable variables $x_j = f_j(\ell_1, \ldots, \ell_{\bar{M}})$, $1 \leq j \leq M$, also follow an Itô process in the form

$$dx_j = df_j(\ell_1, \ldots, \ell_{\bar{M}}) = \sum_{i=1}^{\bar{M}} \left(\mathcal{a}_i \frac{\partial f_j}{\partial \ell_i} + \frac{1}{2} \mathcal{b}_i^2 \frac{\partial^2 f_j}{\partial \ell_i^2} \right) dt + \sum_{i=1}^{\bar{M}} \mathcal{b}_i \frac{\partial f_j}{\partial \ell_i}\, dW_i.$$

Using the Itô process assumption and applying an appropriate rescaling of the features, we will argue that the unknown Jacobian-based distortion that appears in the new distance can be locally estimated by the covariance matrix of the data, i.e., $\mathbf{C} = \mathbf{J}\mathbf{J}^\top$ [Singer and Coifman, 2008].

To do so, we first rewrite dx_j as

$$\begin{aligned} dx_j &= \sum_{i=1}^{\bar{M}} \left(a_i \frac{\partial f_j}{\partial \ell_i} + \frac{1}{2} b_i^2 \frac{\partial^2 f_j}{\partial \ell_i^2} \right) dt + \sum_{i=1}^{\bar{M}} b_i \frac{\partial f_j}{\partial \ell_i} \, dW_i \\ &= \sum_{i=1}^{\bar{M}} A_i^j \, dt + \sum_{i=1}^{\bar{M}} B_i^j \, dW_i \end{aligned}$$

and consider the approximation $x_j(t+\Delta t) - x_j(t) \simeq \Delta x_j$. Thus, we have

$$x_j(t+\Delta t) - x_j(t) \simeq \sum_{i=1}^{\bar{M}} A_i^j \, \Delta t + \sum_{i=1}^{\bar{M}} B_i^j \, \Delta W_i,$$

which we use to compute what we may call the instantaneous covariance between two features in the observable space

$$\begin{aligned} \mathbf{C}_{jk} &= \lim_{\Delta t \to 0} \frac{\mathbf{C}_W \left(x_j(t+\Delta t) - x_j(t), x_k(t+\Delta t) - x_k(t) \mid \mathbf{x}(t) = \mathbf{x}' \right)}{\Delta t} \\ &= \lim_{\Delta t \to 0} \frac{\mathbf{C}_W \left(\Delta x_j, \Delta x_k \mid \mathbf{x}(t) = \mathbf{x}' \right)}{\Delta t} \\ &\simeq \lim_{\Delta t \to 0} \frac{\mathbf{C}_W \left(\sum_{i=1}^{\bar{M}} A_i^j \, \Delta t + \sum_{i=1}^{\bar{M}} B_i^j \, \Delta W_i, \, \sum_{i=1}^{\bar{M}} A_i^k \, \Delta t + \sum_{i=1}^{\bar{M}} B_i^k \, \Delta W_i \mid \mathbf{x}(t) = \mathbf{x}' \right)}{\Delta t} \\ &= \lim_{\Delta t \to 0} \frac{\mathbf{C}_W \left(\sum_{i=1}^{\bar{M}} B_i^j \, \Delta W_i, \sum_{i=1}^{\bar{M}} B_i^k \, \Delta W_i \right)}{\Delta t} \\ &= \lim_{\Delta t \to 0} \frac{\mathrm{E}_W \left(\sum_{i,\ell} B_i^j B_\ell^k \, \Delta W_i \Delta W_\ell \right)}{\Delta t} \\ &= \lim_{\Delta t \to 0} \frac{\Delta t \sum_i B_i^j B_i^k}{\Delta t} = \sum_{i=1}^{\bar{M}} (b_i)^2 \frac{\partial f_j}{\partial l_i} \frac{\partial f_k}{\partial l_i}. \end{aligned}$$

We have arrived then to the matrix expression $\mathbf{C} = \mathbf{J}\mathbf{B}^2\mathbf{J}^\top$. In the preceding we have used that the terms which are independent of Brownian motion W do not contribute to the covariance and that the values of the first two moments of this stochastic process are $\mathrm{E}_W\left(W(t)\right) = 0$ and $\mathrm{E}_W\left(W^2(t)\right) = t$.

In order to remove the dependence on the unknown B matrix, we would like that our latent variables follow an Itô process of the form

$$d\ell_i = \tilde{a}_i \, dt + \mathbf{I} \, dW_i.$$

Following Singer and Coifman [2008], this can be achieved by rescaling our initial latent variables ℓ so they have no dependence on the noise term. In more detail, assume we have $h_i(\ell_i) = \tilde{\ell}_i$ for a general function h, and applying Itô's lemma (Theorem 4.1) we would have

$$dh_i(\ell_i) = \left(a_i \frac{\partial h_i}{\partial \ell_i} + \frac{b_i^2}{2} \frac{\partial^2 h_i}{\partial \ell_i^2} \right) \, dt + b_i \frac{\partial h_i}{\partial \ell_i} \, dW_i.$$

Thus, if we apply a one-dimensional scaling transformation that satisfies $\frac{\partial h_i}{\partial \ell_i} = \frac{1}{b_i}$ to each one of the original inaccessible variables, we arrive to the new latent variables $\tilde{\ell}_i$ that verify

$$\tilde{\ell}_i = dh_i(\ell_i) = \tilde{a}_i \, dt + \, dW_i,$$

for a suitable $\tilde{a}$. After this rescaling and with a slight abuse of notation, we obtain that the covariance matrix on the observable space is locally equivalent to the unknown distortion of the function f, that is

$$\mathbf{C} = \mathbf{J}\mathbf{J}^\top.$$

While the instantaneous covariance $\mathbf{C}$ is associated to the Itô process, we may relate it to the local covariance matrix between two points that are near. In other words, we may assume that the image of a small ball in the latent space is mapped to a small ellipsoid in observable space with the distortion captured by the sample's local covariance which, in turn, suggests to estimate the local Jacobian distortion using this covariance, that is,

$$\mathbf{J}\mathbf{J}^\top(c) \simeq \mathbf{C}(x_i, x_j | \mathbf{x} \in f(\mathscr{B}_c)).$$

More precisely, for points $x^{(p)} \in \mathscr{S}$, we can approximate the distance between the latent variables as

$$\begin{aligned} \|\boldsymbol{\ell}^{(p)} - \boldsymbol{\ell}^{(q)}\|^2 &\simeq \frac{1}{2}(\mathbf{x}^{(p)} - \mathbf{x}^{(q)})^\top [(\mathbf{J}\mathbf{J}^\top)^{-1}(\mathbf{x}^{(p)}) + (\mathbf{J}\mathbf{J}^\top)^{-1}(\mathbf{x}^{(q)})](\mathbf{x}^{(p)} - \mathbf{x}^{(q)}) \\ &= \frac{1}{2}(\mathbf{x}^{(p)} - \mathbf{x}^{(q)})^\top [\mathbf{C}^{-1}(\mathbf{x}^{(p)}) + \mathbf{C}^{-1}(\mathbf{x}^{(q)})](\mathbf{x}^{(p)} - \mathbf{x}^{(q)}). \end{aligned}$$

This suggests that, in order to estimate $\mathbf{C}$, we use local sample covariances $\mathbf{C}\left(\mathbf{x}^{(p)}\right)$ that can be estimated using point clouds $\mathscr{C}_{\mathbf{x}^{(p)}}$ around $\mathbf{x}^{(p)}$.

We will call this Mahalanobis-like distance the anisotropic or AD distance and we will denote it as

$$\mathcal{D}_{\text{AD}}(\mathbf{x}^{(p)}, \mathbf{x}^{(q)}) = \frac{1}{2}(\mathbf{x}^{(p)} - \mathbf{x}^{(q)})^\top [\mathbf{C}^{-1}(\mathbf{x}^{(p)}) + \mathbf{C}^{-1}(\mathbf{x}^{(q)})](\mathbf{x}^{(p)} - \mathbf{x}^{(q)}). \tag{4.2}$$

At the end, the important fact is that the Euclidean distance $\|\boldsymbol{\ell}^{(p)} - \boldsymbol{\ell}^{(q)}\|$ in the latent variable space can be approximated by this anisotropic distance.

Note that the observable covariance matrix is a $M \times M$ semi-positive matrix but its rank is $\bar{M} \ll M$, with $\bar{M}$ the dimension of the tangent space where the data points are supposed to lie. This fact means that the $\bar{M}$ largest eigenvalues represent the square of the semi-principal axes of the projected ellipsoid and their corresponding eigenvectors are the principal components, while the remaining eigenvalues are zero (up to the noise). Because of this rank property, we can not compute the inverse of the covariance matrix, so what we actually mean with the expression

ALGORITHM 4.1: Anisotropic Diffusion Algorithm.

Input: $\mathscr{S} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$, the original data ordered by some feature ; Parameters: cL, σ ;

Output: $\mathbf{\Psi}_t(\mathbf{x})$, the embedded data set ;

1: Build $\mathscr{C}_{\mathbf{x}^{(p)}}\ \forall \mathbf{x}^{(p)} \in \mathscr{S}$ of size cL ; ▶ *Local Clouds.*

2: $\mathbf{C}\left(\mathbf{x}^{(p)}\right) = \mathbf{C}\left(\mathscr{C}_{\mathbf{x}^{(p)}}\right)\ \forall \mathbf{x}^{(p)} \in \mathscr{S}$; ▶ *Covariance Matrix.*

3: $\mathcal{D}_{\text{AD}}(\mathbf{x}^{(p)}, \mathbf{x}^{(q)}) = (\mathbf{x}^{(p)} - \mathbf{x}^{(q)})^{\top}[\mathbf{C}^{-1}(\mathbf{x}^{(p)}) + \mathbf{C}^{-1}(\mathbf{x}^{(q)})](\mathbf{x}^{(p)} - \mathbf{x}^{(q)})$; ▶ *Local Mahalanobis Distance.*

4: $k_{pq} = e^{\frac{-\mathcal{D}_{\text{AD}}(\mathbf{x}^{(p)}, \mathbf{x}^{(q)})}{\sigma}}$; ▶ *Anisotropic Diffusion Kernel.*

5: Normalize $\mathbf{K}$;

6: Eigenvalues $\{\lambda_r\}_{r \geqslant 0}$ and eigenfunctions $\{\psi_r\}_{r \geqslant 0}$; ▶ *Eigendecomposition of* $\mathbf{K}$.

7: $\mathbf{\Psi}_t(\mathbf{x}) = \begin{pmatrix} \lambda_1^t \psi_1(\mathbf{x}) \\ \vdots \\ \lambda_{\bar{M}}^t \psi_{\bar{M}}(\mathbf{x}) \end{pmatrix}$; ▶ *Anisotropic Coordinates.*

$\mathbf{C}^{-1}$ is the pseudo-inverse computation of the covariance matrix on the $\bar{M}$-dimensional subspace of the principal components [Singer and Coifman, 2008].

Once we have computed $\mathcal{D}_{\text{AD}}(\mathbf{x}^{(p)}, \mathbf{x}^{(q)})$ we can define an anisotropic similarity matrix $k_{pq} = e^{\frac{-\mathcal{D}_{\text{AD}}(\mathbf{x}^{(p)}, \mathbf{x}^{(q)})}{\sigma}}$ in the observable space and apply the same steps of the isotropic procedure that are used to compute the DM projections. This involves the kernel matrix normalization in a row stochastic way, as done in Section 3.2, and the posterior eigenanalysis of this normalized kernel. In this way we can apply the same dimensionality reduction that in DM and define new anisotropic diffusion coordinates that represent the original latent features.

Moreover, in this case the infinitesimal generator of the defined Markov chain coincides with the backward Fokker–Planck operator [Singer and Coifman, 2008] guaranteeing a perfect decoupling of the latent variables which will be represented by the diffusion coordinates. Because of this, it is argued in [Singer and Coifman, 2008] that this method yields non-linear Independent Component Analysis (ICA) coordinates when the latent variables follow an Itô process

The steps to compute this embedded method are summarized in Algorithm 4.1.

4.3 Local NN Regression using AD

Nearest Neighbors (NN) regression is among the simplest local regression methods, and consist in searching κ neighbors and averaging their target values to get a prediction (see [Hastie et al.,

ALGORITHM 4.2: AD-based κ-NN Algorithm.

Input: $\mathscr{S} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$, the original data set ; $\{y_1, \ldots, y_N\}$, their targets ; Parameters: c_L, κ ;

Output: $\hat{\mathbf{y}}$;

1: **for** $\mathbf{x}^{(p)} \in \mathscr{S}$ **do**

2: Build $\mathscr{C}_{\mathbf{x}^{(p)}}$ of size c_L ; ▶ *Local clouds.*

3: $\mathbf{C}\left(\mathbf{x}^{(p)}\right) = \mathbf{C}\left(\mathscr{C}_{\mathbf{x}^{(p)}}\right)$; ▶ *Covariance Matrix.*

4: $\mathcal{D}_{\text{AD}}(\mathbf{x}^{(p)}, \mathbf{x}^{(q)}) = (\mathbf{x}^{(p)} - \mathbf{x}^{(q)})^{\top}[\mathbf{C}^{-1}(\mathbf{x}^{(p)}) + \mathbf{C}^{-1}(\mathbf{x}^{(q)})](\mathbf{x}^{(p)} - \mathbf{x}^{(q)})$;

5: $\mathscr{N}_\kappa = \mathcal{NN}(\mathscr{S}, \mathbf{x}^{(p)}, \kappa)$; ▶ *κ-NN search using $\mathcal{D}_{AD}(\mathbf{x}^{(p)}, \mathbf{x}^{(q)})$.*

6: $\hat{y}_p = \sum_{j \in \mathscr{N}_\kappa} y_q \mathcal{D}_{\text{AD}}(\mathbf{x}^{(p)}, \mathbf{x}^{(q)})$; ▶ *Prediction.*

7: **end for**

2008, Chapter 2.3.2]). We can very easily extend this framework to the AD setting taking advantage of the good property explained above: the AD allows to directly compute distances on the sample manifold that are equivalent to Euclidean distances over the inaccessible latent space using Equation (4.2).

The idea shall be to build models over the unknown parametric features directly on the observable features, without needing to be at the parametric space, and then without having to go through any costly eigenanalysis. In this line, we will define a κ-NN search using the local Mahalanobis distances that approximates the Euclidean distances between the latent variables. Thus, measuring distances over the observable features we will be able to find points that are similar in terms of their latent variables.

Another advantage of a method of this kind will be that, while in the traditional use of Diffusion Methods the embedding obtained cannot be directly applied to new, unseen patterns, in the anisotropic approximation the out-of-sample extension will not be needed as we do not need the embedded coordinates.

A possible, general algorithm to apply this theory is presented in Algorithm 4.2.

The main difficulty in this kind of methods lies on how to define a good local cloud for each point. First of all, we will need to have some order in our data to determine neighborhoods. This type of algorithms works very well for example in temporal series. Of course, this is just a desirable characteristic, as we can always fix the neighbors of a point as the nearest ones in Euclidean—or any other—distance.

Depending on the problem, we will need to define this cloud $\mathscr{C}_{\mathbf{x}^{(p)}}$ in relation with its reference point $\mathbf{x}^{(p)}$ in a different way. A typical option will be to define the cloud around $\mathbf{x}^{(p)}$ but, for example, $\mathbf{x}^{(p)}$ could be the extreme of the interval that conform the set. Moreover, when working

with new, unseen patterns, it could be impossible to define a cloud around the new point $\mathbf{x}$ if we do not know its relative position with respect to the training points. In this case we can approximate the local Mahalanobis distance as

$$(\mathbf{x}^{(p)} - \mathbf{x})^\top [C^{-1}(\mathbf{x}^{(p)})](\mathbf{x}^{(p)} - \mathbf{x}) \quad \forall \mathbf{x}^{(p)} \in \mathscr{D}_{\text{tr}}. \tag{4.3}$$

Anyway, sometimes we will be just interested on used an approximate distance even if we can set clouds around test points just for computational cost reduction. In this case, an alternative distance is defined by

$$(\mathbf{x}^{(p)} - \mathbf{x})^\top [C^{-1}(\mathbf{x})](\mathbf{x}^{(p)} - \mathbf{x}) \quad \forall \mathbf{x}^{(p)} \in \mathscr{D}_{\text{tr}}. \tag{4.4}$$

Even though the "real" Mahalanobis distance should yield better results, these approximations make sense as we are building a neighborhood also based on the unknown distortion of the parametric function, but just seen from the training points of view in Equation (4.3) or with the new point as reference, as shown in Equation (4.4).

4.3.1 Possible Applications of this Theory

The AD classical method has been recently used for different applications. For example, Kushnir et al. [2012] used it for classifying earth structures, and in Talmon et al. [2011] and Talmon et al. [2013] DM and AD were used for locating a source in a reverberant room using measurements from a single microphone. But in all these examples, the work is done over the anisotropic diffusion embedding.

The important advantage introduced in Section 4.3 is that we no longer need a costly eigen-analysis to capture the manifold's metric since we can approximate Euclidean distance in latent space directly on the sample manifold $\mathscr{M}$ using Equation (4.2).

The previous set up applies naturally to data that have an underlying temporal structure, such as meteorological phenomena. But it can be applied not only over temporal series but also over data that can be one-dimensionally ordered in some way, for instance, using a one-dimensional target in the training set. In this thesis we will follow the approach presented in Section 4.3 in the next two sections, where we will modify Algorithm 4.2 to deal with two concrete problems, Computed Tomography (CT) scan images location and wind ramps detection. As we shall see, our procedures have in both cases a good performance.

4.4 AD for Location Prediction in CT Scan Images

4.4.1 Problem Definition

A general problem in location-dependent data sets is data misalignment, usually due to different sources of information, such as different experiments done in different periods of time, under different conditions, etc. In particular, this situation arises in medical data sets when data come from different patients or represent the evolution of a concrete patient whose conditions have changed over time.

More concretely, we deal here with the problem of labeling CT scan image slices according to their positions on the human body, which is a key issue for studies where a large number of images is involved. For instance, in medical research it is interesting to analyze and compare as many patients as possible or, alternatively, as many tomographies from the same patient as possible, so the availability of a large image database is highly desirable. However, with the development of new technologies and equipment, image databases have also become very large in size requiring more than 1GB for just a single full body CT scan. On the other hand, the acquired images are typically numbered from head to foot, but not necessarily on a comparable manner as the distance between two tomographies can differ from set to set, being typical resolutions 0.5mm, 1mm or 1.5mm. This, combined with the huge data sizes associated to CT scan images, can result in a problem when radiologists need to use them.

A good example of the need of labeled CT scan images is the analysis of the effectiveness of some chemotherapeutic agents, looking for reductions of neoplastic mass. In these clinical trials, normally the object of the study is a concrete zone, usually very small, especially in relation to the whole area that covers each scan. Imagine, for example, that we were studying pancreatic cancer. The object of the study would be a well localized and relatively small organ, and we could be working with about 50 different patients, all of them with their own non-labeled CT scan images. To localize the patients' pancreas among all the CT scan images can be a tedious, time-consuming task for the physician if he has to organize them by hand.

The labeling of CT scan images could be also used for diagnostic purposes to automatically retrieve a concrete area from several sets of CT scan images that were taken at different times. In this case, the advantage will be more limited as the time spent looking for a concrete area in a given patient is much smaller.

It is thus very interesting to be able to automatically locate and classify large numbers of CT scan images and this problem has recently received increasing attention, although it has passed largely unnoticed by the Machine Learning (ML) community. Our starting point will be the work in Emrich et al. [2010] and Graf et al. [2011], where a thorough study has been done in order to determine a good set of features that describes properly CT slice images and that

can be used to locate each slice's position value in the normalized range $[0, 180]$. These papers can be considered the state-of-the-art for the CT image scan location problem. To find the most suitable position of a CT scan slice, an instance-based regression model can be employed. Currently, the state-of-the-art proposals for this problem are to use a two-stage κ-NN search in terms of Euclidean distances on image features.

These state-of-the-art methods are very robust and easy to implement while it can deal with very heterogeneous samples, such as a CT scan images from different patients. Two steps are done in this method because slices can be similar due to two main reasons: obviously, that they correspond to close body locations but also that the images come from the same patient. Observe that CT scan images are usually done over rather narrowly localized regions and the features associated to moderately separated areas in the same region of a patient may be quite similar. In this problem, we are only interested in the first reason, as the second cause of similarity can distort the prediction.

More precisely, given a new scan to be labeled, the first step looks for a few slices of each patient in the training sample that are closest to it, and the second one searches for the nearest neighbors among the slices selected in the first step. The location of the slice is predicted by the mean of the locations of these second neighbors. This approximation is useful because, as just explained, on the one hand, CT slices taken on concrete body regions not too far apart appear to be more similar to other slices from different locations of the same patient than between those for similar locations of different patients. But, on the other, tomographies from similar locations of different patients give more position information than images from the same patient.

While simple, this method yields rather good results but it leads naturally to question whether the Euclidean metric in scan feature space reflects scan similarities best or there may be other metrics better suited to this problem. As shown in this section, it is possible to improve on the current state-of-the-art by just using a different metric for weighting the selected neighbors that captures the intrinsic geometry of CT scan features better than a simple mean between the nearest points in Euclidean distance.

Diffusion methods are very attractive when looking for relevant neighbors of a given scan whose body location is sought, as the diffusion metric may be more informative than the straight Euclidean metric on the initial CT scan features. As shown in Section 4.2, AD assumes a different underlying data model from DM that makes possible to compute the intrinsic manifold distance directly in the ambient sample coordinates and this metric is just what we need in order to better weighted the nearest neighbor suited to the CT scan features. As we shall discuss, here we have applied the general NN search approach currently used in the state-of-the-art methods for CT scan position labeling but averaging then the locations of the selected tomographies according to an appropriate anisotropic diffusion metric. With the technique proposed, all the images will be autonomously labeled with their location in the human body, so the selection of the acceptable

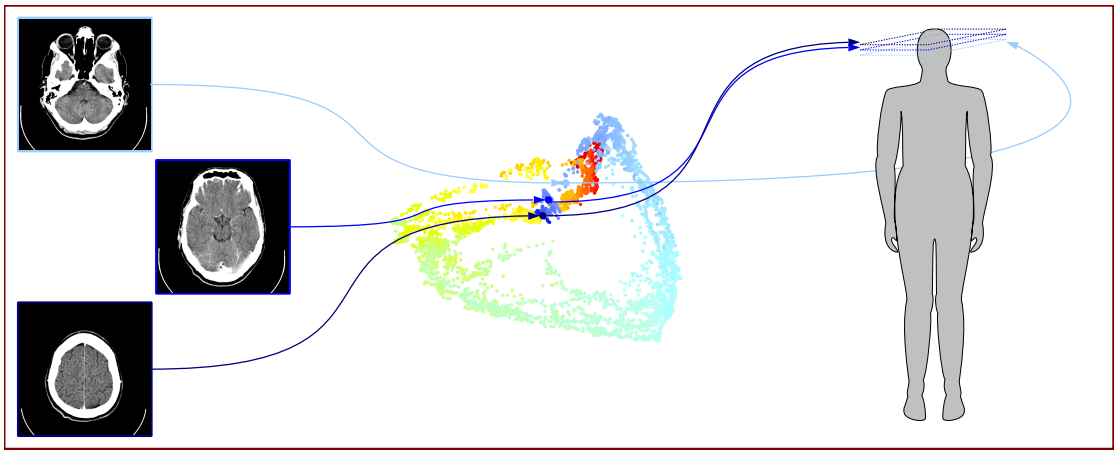


FIGURE 4.2: Graphical definition of the CT scan images location problem and the adopted solution.

images for the study will be automatic, increasing thus efficiency without affecting the quality of the clinical trial.

4.4.2 Methodology

4.4.2.1 Model

As mentioned before, in Emrich et al. [2010] and Graf et al. [2011] it is suggested to use a two-stage κ-NN search in terms of Euclidean distances on image features for CT scan images location. According to this method, given a new test image $\tilde{\mathbf{x}} \in \mathscr{D}_{\text{te}}$, a training sample $\mathscr{D}_{\text{tr}}$ is made of CT slices $x_i^{(p)}$ with known positions in a given sample patient p and the first search looks for the slices nearest to $\tilde{\mathbf{x}}$ inside each patient in the sample, obtaining a scan subset $\mathscr{N}^{(p)}$ with the nearest κ_1 samples in Euclidean distance. The second search is done over the $\cup_p \mathscr{N}^{(p)}$ subset and selects the final κ_2 samples. To ensure a good mixing of tomographies from different sample patients we should take $\kappa_2 > \kappa_1$. The predicted location of $\tilde{\mathbf{x}}$ is given by averaging the locations of the selected nearest CT scan images.

The success of this approach obviously relies on the appropriateness of the Euclidean metric for neighbor selection. A possible improvement could be derived replacing a straight average of neighbor positions by weighted versions that take into account the Euclidean distances between the image to be positioned and their chosen neighbors. Depending on the concrete choices that are made for the weights, several methods can be considered but all of them fit in the general framework given by Algorithm 4.3. Observe that all the κ-NN searches are done using the Euclidean metric and the AD metric is used to compute the weighting coefficients of the local model.

ALGORITHM 4.3: Two Step κ-NN Location Prediction Algorithm for CT scan images.

Input: $\mathscr{D}_{\text{tr}} = \mathscr{P}^1 \cup \ldots \cup \mathscr{P}^P$, the training sample with $\mathscr{P}^p = \{(x_n^{(p)}, \mathbf{y}_n^{(p)})\}_{n=1}^{N_p}$, a subset of slices from patient p where $\mathbf{y}_i^{(p)}$ is the position of pattern $x_i^{(p)}$; A new test point $\tilde{\mathbf{x}} \in \mathscr{D}_{\text{te}}$;

Output: $\hat{\mathbf{y}}$, the predicted position for $\tilde{\mathbf{x}}$;

1: $\mathscr{P}_{\text{pc}} = PCA(\mathscr{P})$; ▶ *PCA of the original sample.*

2: $\tilde{\mathbf{x}}_{\text{pc}} = PCA(\tilde{\mathbf{x}})$; ▶ *PCA of the test pattern.*

3: **for** $p = 1, \ldots, P$ **do**

4: $\quad \mathscr{N}^{(p)} = \mathcal{NN}(\mathscr{P}^p_{\text{pc}}, \tilde{\mathbf{x}}_{\text{pc}}, \kappa_1)$; ▶ *κ_1 neighbors with respect to patient p.*

5: **end for**

6: $\mathscr{N} = \cup_{p=1}^{P} \mathscr{N}^{(p)}$; ▶ *Union of the previous neighbors.*

7: $\mathscr{NN} = \mathcal{NN}(\mathscr{N}, \tilde{\mathbf{x}}_{\text{pc}}, \kappa_2)$; ▶ *Second search over $\mathscr{N}$.*

8: $\hat{\mathbf{y}} = \sum_{\mathscr{NN}} w_i \mathbf{y}_i$; ▶ *w_i depends on the metric.*

For instance, the state-of-the-art model, which we refer here as $2k\mathcal{NN}$, relies on the two-step κ-NN search using Euclidean pattern distance and gives the position of a new CT slice as the average of the positions of the final nearest neighbors.

A first possible refinement of this basic model is to rate the importance of each neighbor by its distance to the test point and to output then the predicted position as a distance-weighted average of the nearest neighbors. We call this $2k\mathcal{NN}_w$.

Further, in a diffusion context a natural idea when computing the w weights is to replace Euclidean distance by the diffusion distance (or, equivalently, by the Euclidean distance in the diffusion features). However, this is rather costly when large samples are involved and, moreover, we would have to use an appropriate approximation to the diffusion map when working with test patterns. We simplify on this by still selecting the neighbors using Euclidean distance but weighting the nearest neighbors using the diffusion kernel, i.e., using the weights

$$w_i = \frac{e^{-(\|\mathbf{x}^{(i)} - \tilde{\mathbf{x}}\|^2)/\sigma^2}}{\sum_j e^{-(\|\mathbf{x}^{(j)} - \tilde{\mathbf{x}}\|^2)/\sigma^2}}$$

associated to the diffusion similarity matrix. We will denote this as $2k\mathcal{NN}_{\mathcal{DM}}$ although we point out that this is not exactly a diffusion mapping approach.

The same idea can be applied using an anisotropic diffusion kernel, defined by a local Mahalanobis distance in the following way:

$$w_i = \frac{e^{-(\mathbf{x}^{(i)} - \tilde{\mathbf{x}})^\top \mathbf{C}_i^{-1} (\mathbf{x}^{(i)} - \tilde{\mathbf{x}})/\sigma}}{\sum_j e^{-(\mathbf{x}^{(j)} - \tilde{\mathbf{x}})^\top \mathbf{C}_j^{-1} (\mathbf{x}^{(j)} - \tilde{\mathbf{x}})/\sigma}},$$

TABLE 4.1: Weights assigned to each different model

Models	**kNN Distance**	**Weights**
$2k\mathcal{NN}$	Euclidean	$w_i = \frac{1}{\kappa_2}$
$2k\mathcal{NN}_w$	Euclidean	$w_i = \frac{\|\mathbf{x}^{(i)} - \tilde{\mathbf{x}}\|^{-1}}{\sum_j \|\mathbf{x}^{(j)} - \tilde{\mathbf{x}}\|^{-1}}$
$2k\mathcal{NN}_{\mathcal{DM}}$	Euclidean	$w_i = \frac{e^{-(\|\mathbf{x}^{(i)} - \tilde{\mathbf{x}}\|^2)/\sigma^2}}{\sum_j e^{-(\|\mathbf{x}^{(j)} - \tilde{\mathbf{x}}\|^2/\sigma^2}}$
$2k\mathcal{NN}_{\mathcal{AD}}$	Euclidean	$w_i = \frac{e^{-(\mathbf{x}^{(i)} - \tilde{\mathbf{x}})^\top \mathbf{C}_i^{-1} (\mathbf{x}^{(i)} - \tilde{\mathbf{x}})/\sigma}}{\sum_j e^{-(\mathbf{x}^{(j)} - \tilde{\mathbf{x}})^\top \mathbf{C}_j^{-1} (\mathbf{x}^{(j)} - \tilde{\mathbf{x}})/\sigma}}$

with $\mathbf{C}_i = \mathbf{C}(\mathbf{x}^{(i)})$ the covariance matrix of a symmetric cloud $\mathscr{C}_{\mathbf{x}^{(i)}}$ of cL points near in location to $\mathbf{x}^{(i)}$ which will be the center of the cloud, thus $\mathscr{C}_{\mathbf{x}^{(i)}} = \{\mathbf{x}^{(i-cL/2)} \cdots \mathbf{x}^{(i)} \cdots \mathbf{x}^{(i+cL/2)}\}$.

Note that in this approach we are not applying the real distance approximation Equation (4.2) and only the covariance matrices for each training point $\mathbf{x}^{(i)}$ are computed. This is so because the covariance matrix for the test point cannot be computed in this way, as its location is unknown. In any case, as explained before, computing the weights using only the covariance matrix relative to the training points is justified because this distance measures how similar the test point is to each training point (training points will be the reference in this case). In other words, if $\mathbf{x}^{(i)}$ is close to $\tilde{\mathbf{x}}$ we could expect $\mathbf{C}(\mathbf{x}^{(i)}) \simeq \mathbf{C}(\tilde{\mathbf{x}})$ while this will not be so if $\mathbf{x}^{(i)}$ and $\tilde{\mathbf{x}}$ are far apart. The combination of the Euclidean two-step κ-NN search and the AD kernel weights will be denoted as $2k\mathcal{NN}_{\mathcal{AD}}$.

Table 4.1 summarizes the weight choices for the base location methods and the extensions discussed.

4.4.2.2 Some Practical Issues

Next, we address some practical issues that should be taken into account when the previous methods are applied.

Number κ of Nearest Neighbors. Recall that to apply our two step κ-NN search we have to set the values of two parameters: κ_1 and κ_2. For this concrete problem, to avoid the mixing of tomographies from different patients, Graf et al. [2011] suggests that $\kappa_2 > \kappa_1$.

To define κ_1, the original method of Emrich et al. [2010] and Graf et al. [2011] suggests to use a small value in between 2 and 5. We have examined the performance of alternative choices in the range $2 \leq \kappa_1 \leq \kappa_2$, as the value of κ_2, if fixed beforehand, is obviously a limit for κ_1. As we will see in Figure 4.4, model errors stabilize as κ_1 grows and the results are more robust specially for $2k\mathcal{NN}_{\mathcal{AD}}$ model.

For κ_2 we have followed an approach that takes into account the sample's density, as we should take a larger number of slices from a dense sample and a smaller number from a sparser one. To determine its value we have computed, in the training set, the average number of neighbors with a location-distance to each training pattern smaller than a certain quantity. This location-distance, that will define the radius of the neighborhood, should be big enough to ensure that more than one neighbor will be chosen, but small enough in order to ensure that the neighborhood is formed by images located almost in the same position than the reference one. Appropriate radii for the neighborhood will be values κ of 0.05cm or 0.1cm which correspond to a 0.03% and a 0.06% respectively of the maximum distance (180), that represents an almost negligible minimum error in an ideal case. These quantities also make sense because the mean separation location-distance for the training points is 0.017cm, which means that taking a neighborhood radius of at most 0.1cm we are ensuring a moderate number of neighbors, greater than one but not too big. This would ensure that the κ-NN model will be robust but without having a big error.

σ Parameter for the Diffusion Kernels. The parameter σ of the diffusion kernels determines the size of the "effective" neighborhood of each pattern, and for this particular problem, we are interested on ensuring a moderately small neighborhood so a small percentile of the distances should be taken into account (see Section 3.2). To define this parameter the distance has to be in accordance with the metric of the method, which means that for DM the distance will be Euclidean and for AD the local Mahalanobis distance is used.

Principal Component Analysis. An issue specific to the AD approach is the need to work with regular covariance matrices, as they have to be inverted. However, given the relatively small size of the subsamples to be used for covariance computation, the high dimensionality of the pattern features would result in singular covariances. This problem has also arisen in the CT scan location literature and, as done there, we will address it through Principal Component Analysis (PCA).

The PCA features are selected by sorting the eigenvalues of the full training sample covariance matrix in descending order and choosing a number of projected components so that a given variance percentage var is retained. Figure 4.3 depicts in logarithmic scale on the y axis how the marginal variance (or, equivalently, the eigenvalues) decays as more components are considered. It can be seen that the number of components needed to explain 70% of variance is about twice the number needed to explain 60% and, moreover, the number approximately doubles again if we want to explain a 80%. From 80% on, the number of the required projected components starts growing fast and dimensionality reduction is poor. Because of this, we will work in our

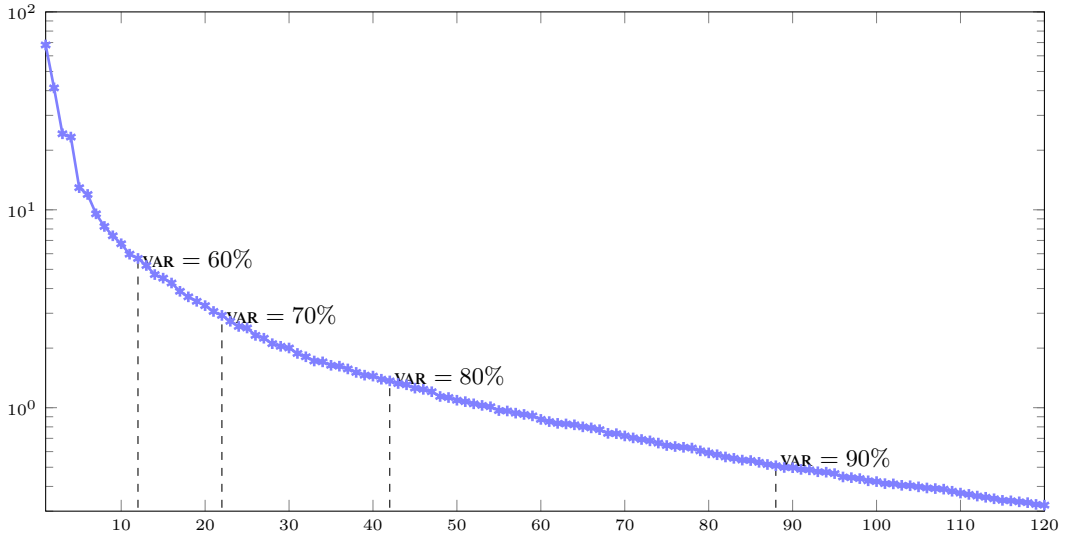


FIGURE 4.3: First 120 PCA eigenvalues for one of the training sets in descending order and logarithmic scale.

experiments with PCA projections that explain 60%, 70% and 80% of variance, and analyze the difference in their results.

Cloud Size for Covariance Estimates. Another AD-specific issue is the need to compute the local covariance estimates $\mathbf{C}_i$ around each training point $\mathbf{x}^{(i)}$ and, therefore, to define the associated point clouds and the metrics used to do so.

To decide how to select the nearest points to a given $\mathbf{x}^{(i)}$ to form its covariance cloud we have two options. The first one is to work with patient-determined clouds, made up of the points closest to $\mathbf{x}^{(i)}$ from the patient's image. The alternative is to work with location clouds, made up of only training sample tomographies that have a similar location to that of $\mathbf{x}^{(i)}$, without regarding the patient they belong to. Observe that we can follow this second option because we only define clouds for the training patterns, whose locations are known. This second approach is much simpler and, as we shall see, gives good results.

We have also to decide the size of the cloud used for the local covariance estimates $\mathbf{C}_i$. Loosely speaking, a "good" cloud size should be the biggest cloud whose points maintain a tangent space close to that at $\mathbf{x}^{(i)}$ and since we are interested in the location of the different scan images, these tangent spaces should be one-dimensional. Since tangent space approximation can only be very local, this suggests to work with moderate cloud sizes. On the other hand, cloud size must be bigger than the dimension of the PCA projection so that the local covariance matrix is not singular. As a reasonable compromise we will work with clouds for which we select a number $2 \times$ projected dimension of points from the global set.

We would like to point out that while there are several parameters that have to be chosen for each algorithm, the above procedures give reasonable prescriptions on how to choose them according to objective procedures that avoid costly explorations of parameter space and still give good results, as we describe next.

4.4.3 Experimental Results

4.4.3.1 Data Description

The data used was retrieved from a set of $53,500$ CT images from 97 different patients in the UCI repository [Frank and Asuncion, 2010]. The average number of images per patient is 551.54, with a minimum of 66 and a maximum of $1,749$. In this repository, each CT slice has been pre-processed, being first scaled to a common resolution and then its features have been described by two histograms in polar space. The first histogram captures the location of bone structures in the image, the second one represents the location of air inclusions inside the body and, as the information of both histogram is complementary, they are concatenated to form a final feature vector of 384 attributes. The target variable is the relative location of an image on the axial axis. For this sample, the target has been computed in two steps: first, up to 10 different distinct landmarks in each CT scan with known locations have been manually annotated and then, the location of slices in between the previous landmarks was interpolated. Location values were set in the range $[0, 180]$ where 0 corresponds to the top of the head and 180 to the feet soles.

For our experiments we will build 9 training sets using all the images corresponding to 10 consecutive patients, selecting then patients $1-10$ as the first training set, $11-20$ as the second one and so on until patients $81-90$. We exclude the training set made of patients $91-97$ since they form a subset with less number of images than the other subsets, that have about $5,000$ images each. To build the test set associated to each training one, we randomly select 100 images from each one of the 87 patients not included in the corresponding training set; thus, each test set has $8,700$ patterns.

4.4.3.2 Parameter Setting

We establish some suitable values for the different parameters following the guidelines summarized in Table 4.2.

We will focus first our attention on the common parameters for all the methods, which are κ_1, κ_2 and the variance explained vAR by the PCA coordinates. In order to determine the behavior of these parameters and the dependence of the methods to the different values, we have made different experiments using all the possible, consistent values for them. This means that we

TABLE 4.2: Parameters involved in our experiments and guidelines used to select their optimal values.

Parameter	**Selecting Range**
VAR	$\{60\%, 70\%, 80\%\}$
κ_1	$[2, \max(\kappa_2)]$
κ_2	$r = \{0.05\text{cm}, 0.1\text{cm}\}$
σ	$\rho = 10\%$
CL	$2 \times$ PCA-projected dimension

TABLE 4.3: κ_2 values selected for the two neighborhood radius by the density estimation method.

r	$01-10$	$11-20$	$21-30$	$31-40$	$41-50$	$51-60$	$61-70$	$71-80$	$81-90$
0.05	8	8	9	9	9	7	6	6	9
0.1	17	16	18	18	19	13	13	13	18

try three different variances explained (60%, 70% and 80%). We will work with neighborhood radius values r of 0.05 and 0.1 to determine κ_2 according to the sample density method described above and, since κ_2 depends on the training set, it will have slightly different values for the nine experiments. For κ_1 we explore the range $[2, \max(\kappa_2)]$, with the maximum taken over the training sample dependent κ_2 values (recall that κ_1 is obviously bounded by the choice of κ_2). These values for the different neighborhood radii are shown in Table 4.3. As the selection of κ_1 and κ_2 neither depends on the method applied nor on the other parameter values, the same κ-values will be used for all the models.

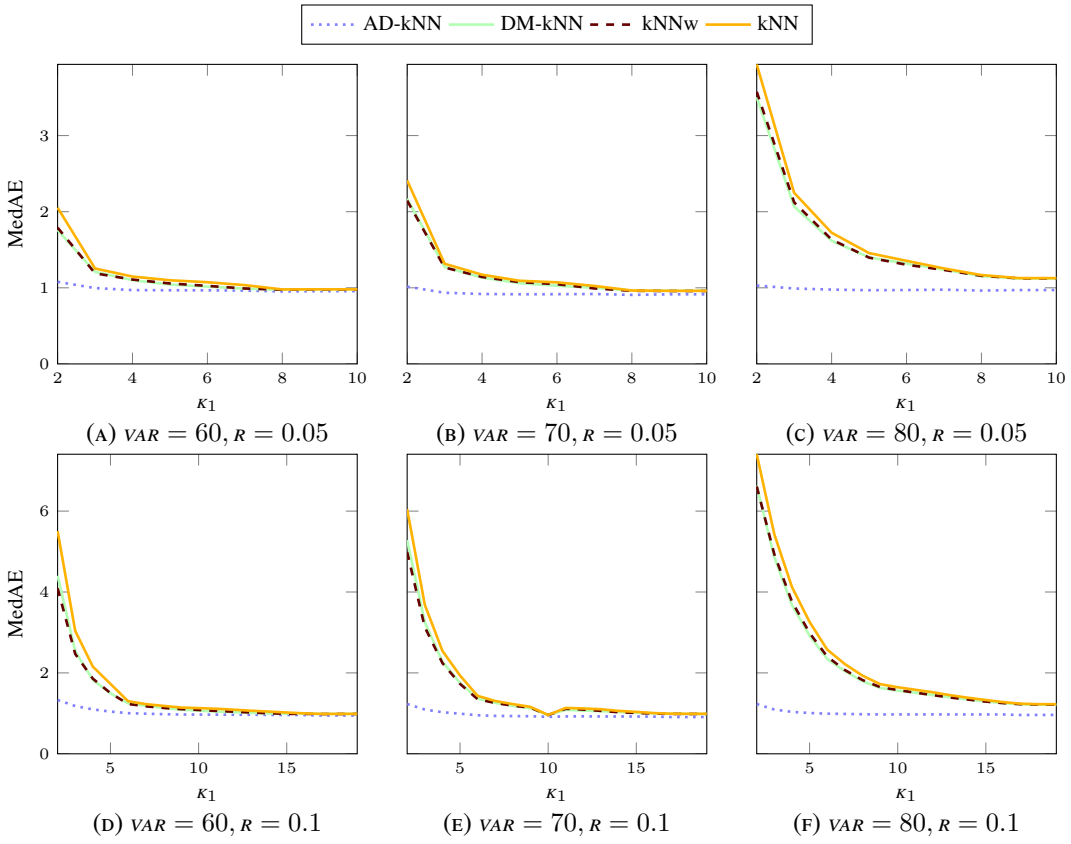
The values of the AD specific parameters depend on the training sample having thus different values for each training subset. But for all of them the same method has been applied: the parameter σ has been fixed as the 10% percentile of the overall sample kernel distances so that the resulting neighborhood will be small. Recall that the parameter CL used to compute the covariance matrix at each training point is taken as twice as many points as those retained PCA-projected dimension, being neither too big to lose the local approximation point of view and nor so small that the matrix will be not singular. Table 4.4 contains the values of these parameters used in the following experiments. The corresponding results, that we discuss next, are shown in Table 4.5.

4.4.3.3 Results

The results of these experiments are shown in Figure 4.4, where each row shows the errors for a different neighborhood radius and each column represents a different variance explained. The y axis in all the graphics presents the average over the 9 training sets of the Median Absolute Error

TABLE 4.4: Values of the $2\mathcal{KNN}_{\mathcal{AD}}$ parameters σ and cL. They correspond to a 70% explained variance, $\kappa_1 = 10$ and the training set dependent values of κ_2 with a radius of 0.1cm.

	$01-10$	$11-20$	$21-30$	$31-40$	$41-50$	$51-60$	$61-70$	$71-80$	$81-90$
σ	2113.7	2723.8	3242.9	3300.1	4746.3	3070.7	3521.4	2097.3	3868.6
cL	44	44	54	56	46	48	42	42	48



(A) $VAR = 60, R = 0.05$ (B) $VAR = 70, R = 0.05$ (C) $VAR = 80, R = 0.05$

(D) $VAR = 60, R = 0.1$ (E) $VAR = 70, R = 0.1$ (F) $VAR = 80, R = 0.1$

FIGURE 4.4: MedAE curves for the different models varying the parameter values. In the first row are shown the results with a neighborhood radius of 0.05cm and in the second row appears the results with a $R = 0.1$cm. The first column corresponds with the experiments done fixing a variance explained of 60%, the second column is the 70%-variance explained and the last one corresponds to the 80%.

(MedAE) computed as percentages of the maximum value of 180 of the normalized distance. We have employed MedAE instead of Mean Squared Error (MSE) because it is more robust with respect to the large variability of the data and the presence of outliers.

As it can be seen, the new models defined here outperform the previously considered κ-NN technique. Moreover, $2\mathcal{KNN}_{\mathcal{AD}}$ is a clear winner for every parameter value and note that in every case the $2\mathcal{KNN}_{\mathcal{AD}}$ error remains essentially stable independently from the values of κ_1, κ_2 or VAR. This good overall behavior hints to a reliable and robust method.

TABLE 4.5: Experimental results for the 9 training sets and its average, executed for $\kappa_1 = 10$, κ_2 determined by a neighborhood of 0.1cm and a PCA-dimension given by a 70% of variance explained. MedAE and MedAD in percentage are shown as error metric for predicting the location of each image slice.

Train. sets	$2k\mathcal{NN}_{\mathcal{AD}}$	$2k\mathcal{NN}_{\mathcal{DM}}$	$2k\mathcal{NN}_{w}$	$2k\mathcal{NN}$
$1-10$	$1.034\% \pm 0.74\%$	$1.036\% \pm 0.75\%$	$1.035\% \pm 0.75\%$	$1.036\% \pm 0.75\%$
$11-20$	$1.051\% \pm 0.74\%$	$1.114\% \pm 0.80\%$	$1.115\% \pm 0.80\%$	$1.113\% \pm 0.80\%$
$21-30$	$0.900\% \pm 0.64\%$	$0.984\% \pm 0.72\%$	$0.984\% \pm 0.72\%$	$0.987\% \pm 0.73\%$
$31-40$	$1.005\% \pm 0.68\%$	$1.028\% \pm 0.72\%$	$1.028\% \pm 0.72\%$	$1.027\% \pm 0.72\%$
$41-50$	$0.861\% \pm 0.58\%$	$0.906\% \pm 0.61\%$	$0.901\% \pm 0.61\%$	$0.906\% \pm 0.61\%$
$51-60$	$0.799\% \pm 0.56\%$	$0.879\% \pm 0.64\%$	$0.874\% \pm 0.63\%$	$0.880\% \pm 0.64\%$
$61-70$	$0.837\% \pm 0.58\%$	$0.858\% \pm 0.60\%$	$0.856\% \pm 0.60\%$	$0.858\% \pm 0.60\%$
$71-80$	$0.861\% \pm 0.58\%$	$0.905\% \pm 0.63\%$	$0.899\% \pm 0.62\%$	$0.906\% \pm 0.63\%$
$81-90$	$0.884\% \pm 0.59\%$	$0.933\% \pm 0.65\%$	$0.930\% \pm 0.65\%$	$0.931\% \pm 0.65\%$
Average	$\mathbf{0.915\% \pm 0.63\%}$	$0.960\% \pm 0.68\%$	$0.958\% \pm 0.68\%$	$0.960\% \pm 0.68\%$

We can also observe in Figure 4.4 that the results tend to improve as the value of κ_1 increases until they stabilize. Observing the dependence to the variance explained, the 60% image shows a slightly higher error. In the 80% case the different models do not converge to the MedAE of the $2k\mathcal{NN}_{\mathcal{AD}}$ model for any κ_1 value. The MedAE values shown in the figure allow us to conclude that the 70% models are slightly better than the others. This can be due to the fact that when a 60% of variance is explained we do not have enough information in the projected space. On the other hand, a 80% of variance explained involves a large number of variables. Recall that in Figure 4.3 (that was shown in logarithmic scale in the y axis) the number of eigenvalues and, hence, PCA dimensions, essentially doubles when going from 60% to 70% of variance and also from 70% to 80%. Since the PCA features are of the form $\mathbf{u} \cdot \mathbf{x}$ with $\mathbf{u}$ a unitary vector, they should have similar magnitudes. Then, when 80% variance is considered, squared Euclidean distances approximately double from the ones obtained with 70% variance and Euclidean pattern distance may be greatly influenced by the extra, less relevant 80% PCA features. The covariance-based Mahalanobis distance appears more robust in this situation, something that may account for the much more stable results obtained by $2k\mathcal{NN}_{\mathcal{AD}}$ for the different variance values.

The good average MedAE results obtained for the $2k\mathcal{NN}_{\mathcal{AD}}$ model also hold for all the individual training sets. This is shown in Table 4.5 that corresponds to models when 70% of variance is explained and presents MedAE and their Median Absolute Deviations (MedAD). At first sight, the MedAD values appear to be rather large with respect to MedAE, but taking into account the high variability present in these data and that a deviation of a 1% correspond to a variation of approximately 1.8cm, these MedAD are reasonable. MedAEs are also given in percentages relatives to the 180 maximum distance. The parameters used in the table are those giving the smallest error for all models, namely $\kappa_1 = 10$ and κ_2 determined by $\kappa = 0.1$cm.

Finally, notice also that, in the $2k\mathcal{NN}_{\mathcal{AD}}$ case, the nearest neighbors are weighted by the

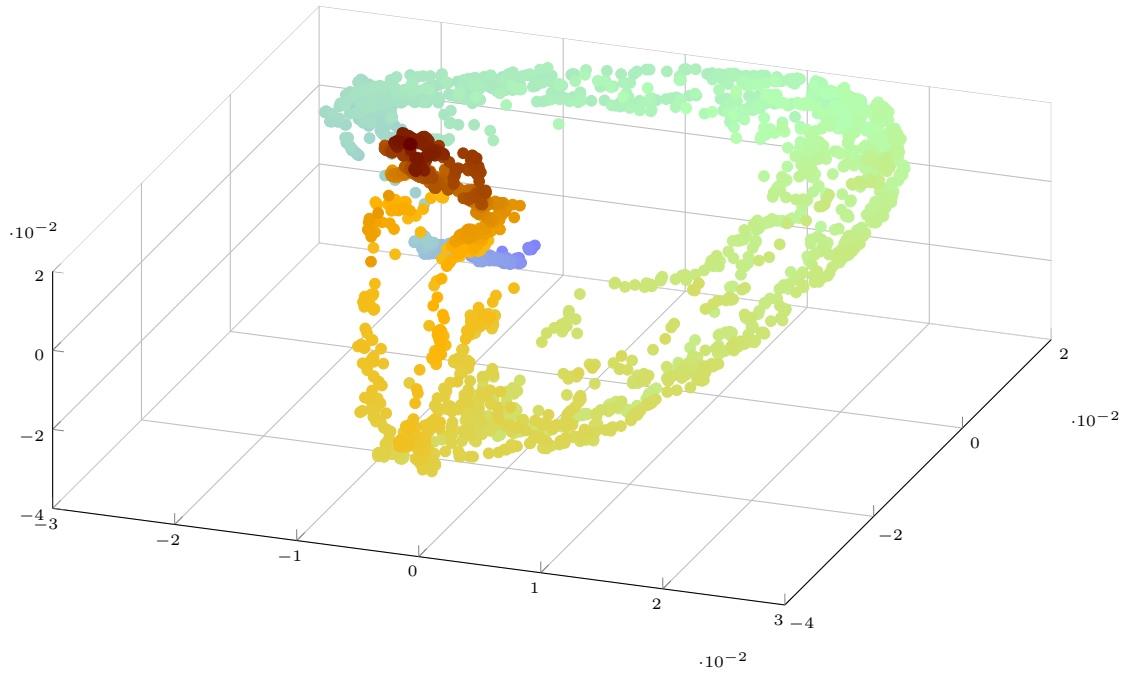


FIGURE 4.5: First two AD embedded coordinates colored by location, where the dark blue color represents the location 0, i.e. the top of the head, and the red color represents the maximum distance that symbolize the feet soles.

anisotropic distance, that we would expect to reflect the Euclidean distance on the underlying latent feature space. If this is true, the diffusion embedding on latent space should also reflect location information. We can see this fact in Figure 4.5 where the first two anisotropic diffusion coordinates of one of the training sets are depicted, colored by location. As seen, colors show a continuous evolution along the diffusion coordinates of the training sample, as to be expected of location variation

In summary, we have shown that anisotropic diffusion methods can be helpful to properly label CT scan images and their computational time does not become a drawback, as they do not need any spectral embedding. Moreover, they can be directly used as predictive tools over new test samples without requiring a special out-of-sample extension. Our results show that the AD averaging of the positions of the κ-NN neighbors clearly outperforms the other methods and yields a more robust technique.

4.5 AD for Wind Power Ramps Detection

4.5.1 Problem Definition

In wind energy, a ramp can be broadly defined as a sudden increase or decrease of energy production that takes place on a short time period. These kind of events are an important factor

in the integration of wind energy in the electrical system, as ramps force system operators to re-balance current load to keep the system's stability. They are also of obvious interest to wind farm operators as the wind surges that accompany ramps can damage wind turbines.

For these reasons, the study of wind ramps is receiving an increasing attention. Although the field is still wide open, a comprehensive review of recent research can be shown in Ferreira et al. [2011] and one can say that there are two basic approaches for wind ramp detection. The first one tries to predict the magnitude of the increments of wind power production using regression models, either deterministic [Zack et al., 2010; Zheng and Kusiak, 2009] or probabilistic [Bossavy et al., 2013a]. In Zheng and Kusiak [2009], feature selection and several data-mining algorithms are combined to study a 10 to 60 minutes ahead power ramp rates, while Zack et al. [2010] present some methodologies to predict between 0 and 6 hours ahead, providing a confidence band for the predictions. In Bossavy et al. [2013a] wind power ramps are characterized with a derivative filtering approach which is then used together with ensemble wind power forecast to provide ramp predictions. Even though these approaches provide more information than the pure detection of the ramp events, the smoothing effect inherent to Numerical Weather Predictions (NWP) and to regression algorithms can make difficult the effective application of these models. On the other side, other studies apply a purely classification approach [Bradford et al., 2010; Greaves et al., 2009] to detect the ramps using some predefined threshold, independently of their specific amplitude. Finally, in Kamath [2010] ramps are studied from a general point of view and in Kamath [2011] they are discussed in relation to load balancing.

Recently some approaches have emerged which combine a direct wind power estimation with classification techniques to determine whether a ramp is taken place. For example, in Alexander [2011] the wind speed is estimated using an AutoRegressive Moving Average (ARMA) model and then converted to wind power using a modified turbine power curve. Finally, a fuzzy logic inference system is applied to estimate the probability of ramp events. Other interesting approach is that of Ferreira et al. [2012], where the ramps are predicted using an adaptive Hidden Markov Model (HMM), which is fed with NWPs of the wind speed. With this information, the model computes the transition probabilities to the states corresponding to ramp events depending on the current state. It also takes into account the emission probabilities of the wind speed, in an intrinsic combination of regression and classification. We will focus our experiments in this line.

It seems clear that ramps are an important problem that falls clearly within the scope of ML methods. Nevertheless, even the characterization of the wind ramps is still an active field, with recent works such as Sevlian and Rajagopal [2012], Bossavy et al. [2013b], Yu et al. [2013] or Haupt et al. [2014], centered on extending the definition of a wind ramp and studying the statistics for the different possible models of these events. There even does not exist yet a standard methodology for the evaluation of ramp detection procedures or an easy way of comparing

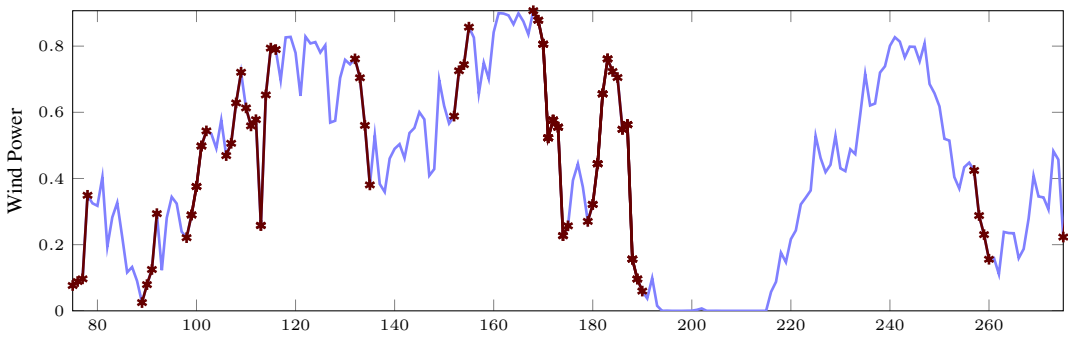


FIGURE 4.6: Selected ramps according to the ramp definition with parameters $\Delta t = 3$ and $\Delta p_{\text{th}} = 0.95$.

results across different works. As a starting point we can mention Barbour et al. [2010], where there are some suggestions to evaluate ramp detection procedures under a classification point of view.

In any case, although there is no standard characterization of wind power ramp, it is an intuitively clear concept, which can be ideally formalized using the derivative of the wind production. Therefore, a ramp will take place in a certain time if the slope of the production p is greater, in absolute value, to a certain predefined threshold, $|p'(t)| \geq \epsilon$, where the absolute value guarantees the detection of both upward and downward ramps. Obviously such a definition approach is not practical, and thus, it has to be approximated [Bossavy et al., 2013a; Ferreira et al., 2011], usually using finite differences instead of derivatives. In this thesis a wind power ramp is defined as a large change in wind production in a relatively short period of time, which is formalized with the condition:

$$|p_{t+\Delta t} - p_t| > \Delta p_{\text{th}}, \tag{4.5}$$

where Δt is the time interval (the "short time period"), Δp_{th} is the power threshold (the "large change"), and t is the starting time of the possible ramp.

The values of Δt and Δp_{th} are critical. Δt, for t measured in hours, should be at least one, and probably even two, as to allow for delays on real time measurements and for load balancing and other actions. Here we will usually work with $\Delta t = 2$, 3 or 4 values. With Δt fixed, the threshold Δp_{th} determines the frequency of occurrence of ramps. If it is too low, there will be many ramp events but most of them of little consequence, whereas if it is too large the ramps will be rare but very relevant. It makes sense to fix the threshold in terms of the maximum extra re-balancing load available or, at a wind farm, on the wind turbine characteristics or operational procedures [Ferreira et al., 2011]. Here we will fix Δp_{th} as the top $\rho\%$ percentile of the magnitude of the increment. Therefore, the probability of $|p_{t+\Delta t} - p_t|$ being larger than Δp_{th} is $\rho\%$. This choice of threshold allows us to consider different ramp settings.

In any case, ramps are sudden, time-localized phenomena, which suggest that ramp forecasting, should rely on real time information. A first approach could be to work in a regression context and to exploit straight time ahead forecasts of wind energy. However, NWP outputs are at best available four times a day and they are likely to have at least a 4–6 hour delay. This suggests complementing this approach using real time information in a now-casting setting. Real time weather measurements are obviously the most sensible option but also require real time data integration, which may be difficult at the system operator level and even at the wind farm level. On the other hand, energy production readings are certainly available and can be used as a proxy of actual meteorological conditions when correcting previous, NWP-only based energy forecasts with those real time readings.

This combination of real time energy readings with short time updates of previous NWP-based forecasts may yield a potentially useful representation of the current energy evolution, to be exploited approaching ramp detection as a classification problem. However, ramp detection is by definition a highly class-unbalanced problem for which global models may not be too successful. The alternative we will follow here is to work locally in an Anisotropic Diffusion setting, finding κ patterns nearest to the current one with respect to the AD metric, computing from them a score that depends on past ramp presence in these patterns and declaring a possible ramp when it goes above a given threshold.

4.5.2 Methodology

Once the wind power ramps are characterized, the objective is to predict, in a certain time τ and for some future horizon H, if a new ramp event is going to start at $\tau + \mathit{H}$. The idea is to define some features that contain at time τ the most relevant information about whether a ramp could begin at time $\tau + \mathit{H}$. These features are collected in the vector $\mathbf{x}^{(\tau)}$.

Using this feature space, we define a score to be used in the classification task that, in this case, will be given by an increase estimation of the wind power production. For this purpose, an appropriate distance between the target pattern and the past ones will be computed to select the κ nearest neighbors which we gather up in a time index $\mathscr{S}_\tau$ set. An approximation to the increase $I_\tau = p_{\tau+\mathit{H}+\Delta t} - p_{\tau+\mathit{H}}$ will be estimated using the corresponding increases of the "near" historical patterns of $\mathscr{S}_\tau$:

$$\hat{I}_\tau = \frac{\sum\limits_{t \in \mathscr{S}_\tau} w_t I_t}{\sum\limits_{t \in \mathscr{S}_\tau} w_t}, \tag{4.6}$$

where $w_t = e^{\frac{-\mathcal{D}_{t,\tau}}{\sigma}}$ is the weight associated to the pattern $\mathbf{x}^{(t)}$ according to its distance $\mathcal{D}_{t,\tau}$ to $\mathbf{x}^{(\tau)}$, and I_t is the historical increase $p_{t+\mathit{H}+\Delta t} - p_{t+\mathit{H}}$. Notice that our definition of the score value $\hat{I}_\tau$ coincide with the prediction of a wind power increase made by a local regression approach

ALGORITHM 4.4: Ramp Events Detection Algorithm.

Input: $p = \{p_1, \ldots, p_\tau\}$, the wind power time series Δt, the ramp duration ;

Output: $\hat{r}_{\tau+H}$, the ramp prediction at time $\tau + H$;

$I_t \leftarrow |p_{t+H+\Delta t} - p_{t+H}|$; ▶ *Increments.*

$\mathbf{x}^{(t)}$ for $t = 1, ..., \tau$; ▶ *Patterns.*

$\mathcal{D}_{t,\tau} = \mathcal{D}(\mathbf{x}^{(t)}, \mathbf{x}^{(\tau)})$; ▶ *Distances.*

$\mathscr{S}_\tau \leftarrow \mathcal{NN}(\mathbf{x}^{(\tau)}, \{\mathbf{x}^{(t)}\}, \kappa)$; ▶ κ *closest patterns to* $\mathbf{x}^{(\tau)}$.

$w_t \leftarrow e^{\frac{-\mathcal{D}_{t,\tau}}{\sigma}}$;

$\hat{I}_\tau \leftarrow \sum_{t \in \mathscr{S}_\tau} w_t I_t / \sum_{t \in \mathscr{S}_\tau} w_t$;

if $\hat{I}_\tau > \Delta \hat{p}_{\text{th}}$ **then**

    $\hat{r}_{\tau+H} = 1$;

else

    $\hat{r}_{\tau+H} = 0$;

end if

based on κ-NN. Nevertheless this forecasting is not good by itself as wind power prediction, specially because of the smoothing effect produced by the neighborhood averaging, but as we shall see, it is still a good starting point for a criterion to detect ramps.

This approach makes the prediction more flexible to the user needs who can fix $\Delta \hat{p}_{\text{th}}$ in order to obtain the desired balance between the number of false and real alarms. The complete method is summarized in Algorithm 4.4.

In order to evaluate the algorithm's performance, we will use Receiver Operator Characteristics (ROC) and Precision–Recall (PR) curves as visualization and evaluation methods. When evaluating a classifier's performance, the following values are often used: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). With these values we can define the *sensitivity*, name Sens $= {}^{\text{TP}}/_{\text{TP}+\text{FN}}$ and the *specificity*, expressed by Spec $= {}^{\text{TN}}/_{\text{TN}+\text{FP}}$, as well as the *precision*, Prec $= {}^{\text{TP}}/_{\text{TP}+\text{FP}}$, that measures the proportion of correct ramp alerts, and the *recall*, which coincides with the sensitivity definition and measures the proportion of ramps correctly detected. ROC curves are the most common method to visualize, evaluate and compare different techniques of binary classification. They show how the number of correctly classified positive examples varies with the number of misclassified negative examples. An alternative to these curves are the PR curves, especially recommended when there exists a large skew in the class distribution [Davis and Goadrich, 2006]. We shall employ both curves to visualize, evaluate, and compare different approaches.

The Area Under the Curve (AUC) value of both curves will be the chosen metric to compare the different methods between them, as it gives a measure of how a model performs under all

the possible requirements of sensitivity-specificity or precision-recall (which are controlled in both cases with $\Delta\hat{p}_{\text{th}}$). This metric will be also employed for selecting the different parameter values involved in each method. To determine whether a difference between AUCs is statistically significant we will apply the DeLong's test [DeLong et al., 1988], which is based on the Mann-Whitney statistic and whose null hypothesis claims that two ROC curves are the same.

4.5.2.1 Pattern Information

A naive first idea to compose the pattern features $\mathbf{x}^{(t)}$ is to use wind power delays and, therefore, work with P-dimensional energy patterns of the form

$$\mathbf{x}^{(t)} = (p_{t-P+1}, \ldots, p_{t-1}, p_t)^\top .$$

While useful for reference purposes, the approximately chaotic medium term wind behavior [Lorenz, 1963] suggests that the auto-correlations between the productions will decrease very fast and such a $\mathbf{x}^{(t)}$ may have a low predictive value.

A possible first improvement of the feature information consists in adding basic daily predictions about the future wind power production derived from NWP forecasts. In our case we will simply apply a Regularized Least Squares (RLS) model as done previously in the experiments of Section 3.4 and formulate in Equation (3.10) and consider the predictions $\{\hat{p}_t^{\text{d}}\}$ that correspond to the hours following p_t and involved in the ramp interval for which we are predicting, i.e.,

$$\mathbf{x}^{(t)} = \left(p_{t-P+1}, \ldots, p_{t-1}, p_t, \hat{p}^{\text{d}}_{t+H}, \ldots, \hat{p}^{\text{d}}_{t+H+\Delta t}\right)^\top .$$

These patterns with prediction information are much more meaningful than those formed only with the wind power delays, but their daily predictions can be further improved using an hourly RLS model that updates them in terms of the last wind power production readings. This is our third option: patterns composed by the P-dimensional delay vectors and the hourly predictions $\{\hat{p}_t^{\text{h}}\}$ for the ramp period, which should capture wind energy evolution around time t better than the daily ones:

$$\mathbf{x}^{(t)} = \left(p_{t-P+1}, \ldots, p_{t-1}, p_t, \hat{p}^{\text{h}}_{t+H}, \ldots, \hat{p}^{\text{h}}_{t+H+\Delta t}\right)^\top .$$

4.5.2.2 Distance Metrics

The most relevant issue in Algorithm 4.4 is the definition of the metric to be used for comparing $\mathbf{x}^{(\tau)}$ with the historical patterns. In this work, the local Mahalanobis AD and Euclidean distances will be used, comparing thus an anisotropic metric with an isotropic one.

The rationale for the use of the AD metric to ramp event prediction is based on the assumption that extreme power ramps correspond to particular values of unknown latent variables that determine wind energy production. More precisely, we consider the just defined wind energy patterns $\mathbf{x}^{(t)}$ that capture the structure of wind production at time t to be determined by unknown latent variable patterns $\boldsymbol{\ell}^{(t)}$. Thus, a possible approach to predict ramps at time τ is to identify previous latent vectors $\boldsymbol{\ell}^{(t)}$, with $t \in \mathscr{S}_\tau$, that are close to the current latent vector $\boldsymbol{\ell}^{(\tau)}$, and then use the corresponding previous wind energy patterns $\mathbf{x}^{(t)}$ and their increments I_t to decide whether or not the current pattern $\mathbf{x}^{(\tau)}$ is associated to a ramp event using Equation (4.6). To use this approach, we must have an estimate of the latent distance $\|\boldsymbol{\ell}^{(\tau)} - \boldsymbol{\ell}^{(t)}\|$, and it is in this context where we can benefit from the AD framework, which allows to define the following approximation

$$\|\boldsymbol{\ell}^{(\tau)} - \boldsymbol{\ell}^{(t)}\|^2 \simeq \mathcal{D}(\mathbf{x}^{(\tau)}, \mathbf{x}^{(t)}) = (\mathbf{x}^{(\tau)} - \mathbf{x}^{(t)})^\top [\mathbf{C}_\tau^{-1} + \mathbf{C}_t^{-1}](\mathbf{x}^{(\tau)} - \mathbf{x}^{(t)}).$$

The previous estimation requires to compute and invert the local covariance matrix $\mathbf{C}_t$ for each possible $\mathbf{x}^{(t)}$, which can be done using a cloud of points around $\mathbf{x}^{(t)}$. To alleviate the possibly large computational cost, we can simplify the Mahalanobis distance relaxing the distance symmetry to

$$\mathcal{D}(\mathbf{x}^{(\tau)}, \mathbf{x}^{(t)}) = (\mathbf{x}^{(\tau)} - \mathbf{x}^{(t)})^\top \mathbf{C}_\tau^{-1} (\mathbf{x}^{(\tau)} - \mathbf{x}^{(t)}), \tag{4.7}$$

thus only one local covariance matrix $\mathbf{C}_\tau$ around the pattern $\mathbf{x}^{(\tau)}$ is needed.

An important issue now is how to select the pattern cloud $\mathscr{C}_\tau$ to compute the covariance matrix $\mathbf{C}_\tau$ around the pattern of time τ. The simplest way is just to work with a time cloud, $\mathscr{C}_\tau = \{\mathbf{x}^{(\tau)}, \mathbf{x}^{(\tau-1)}, \ldots, \mathbf{x}^{(\tau-c_L+1)}\}$, using the c_L patterns closest to $\mathbf{x}^{(\tau)}$ in time.

4.5.3 Experimental Results

4.5.3.1 Models

We will apply two κ-NN models: the standard Euclidean κ-NN ($\mathcal{NN}^{\mathrm{E}}$), and Mahalanobis κ-NN with a time cloud covariance ($\mathcal{NN}^{\mathrm{M_T}}$) using the simplification of Equation (4.7).

We need to distinguish the models according to the pattern information they use. The previous notation will apply to models without predictions, i.e., their patterns only contain wind power delays. An extra D will denote models with daily predictions information: $\mathcal{NN}^{\mathrm{E}}_{\mathrm{D}}$, and $\mathcal{NN}^{\mathrm{M_T}}_{\mathrm{D}}$; and an extra H represents models with hourly predictions information: $\mathcal{NN}^{\mathrm{E}}_{\mathrm{H}}$ and $\mathcal{NN}^{\mathrm{M_T}}_{\mathrm{H}}$.

We first establish some baseline models to define the naive solution to this problem and for helping at the evaluation of the different methods under study. The easiest baseline reference

is a random prediction that assigns at each hour a ramp start with a $\rho\%$ probability. In a ROC curve design, a random model is depicted with a diagonal line, as the random sensitivity and specificity always add up to 100%. In a PR curve the random model appears as a constant given by the percentage of true elements in the sample, in this case at $\rho\%$.

A more competitive baseline model can be obtained using a wind power prediction model, so the ramp alarms will be determined just applying the ramp definition in Equation (4.5) to the predicted increases of size Δt. As this model tends to soften the wind power real curve, different Δp_{th} can be used as thresholds to construct a ROC curve. For this wind power baseline prediction we have chosen again a RLS model using the Tikhonov regularization. For model training we have used as inputs NWPs from the European Center for Medium-Range Weather Forecast (ECMWF) model. We have selected eight meteorological variables: the x and y components of the wind speed and its module, both at surface level and at $100m$ height, the temperature and the pressure. They are given over a $0.25°$ resolution grid. These NWP are available 3-hourly from the hour 0 of the day, and thus the target will be the wind power production for these hours. We call this approach the daily RLS model as we have updated information for each day and it is denoted as $\mathcal{RR}_{\text{D}}$.

An improvement over this model consists in using another hourly RLS model to refine the daily predictions. This model uses as inputs updated information about the real production at each hour. We will train one model per horizon using as inputs the daily wind power prediction for the 6 past hours and their corresponding real productions, and the ahead daily prediction for the horizon that we want to forecast. The target will be the real production at that horizon. This hourly model denoted as $\mathcal{RR}_{\text{H}}$ especially improves the short-term predictions, which are the ones we are interested on for ramp predictions.

4.5.3.2 Local Prediction: Sotavento's Wind Farm

Problem to solve. The previous methods have been applied to the Sotavento's wind farm [Sotavento, 2013], located in the northern Spanish region of Galicia and whose data are publicly available. In particular, the data set used is composed of hourly productions from February 1st, 2010, to January 31st, 2013. The first year will be the training data set, the second year will be used for validation purposes, and the last year will be the test set. For the ramp prediction model, the training data set over which we will look for neighbors will be formed by one year up to the hour before the target hour. Note that all the wind power productions used have been previously normalized by the wind farm nominal power.

We will report our results for ramps with the Δp_{th} of the top 5% and the rather extreme top 1% energy productions. For $\Delta t = 2$, the 5% ramps mean a $\Delta p_{\text{th}} \simeq 20\%$ of the nominal power

TABLE 4.6: Best parameter values of κ, p and cl for each model.

	Parameter	$\mathcal{NN}_{\mathrm{H}}^{\mathrm{M_T}}$		$\mathcal{NN}_{\mathrm{H}}^{\mathrm{E}}$		$\mathcal{NN}_{\mathrm{D}}^{\mathrm{M_T}}$		$\mathcal{NN}_{\mathrm{D}}^{\mathrm{E}}$		$\mathcal{NN}^{\mathrm{M_T}}$		$\mathcal{NN}^{\mathrm{E}}$	
		5%	1%	5%	1%	5%	1%	5%	1%	5%	1%	5%	1%
$\Delta t = 2$	κ	25	23	23	22	23	25	25	25	25	23	25	25
	p	12	12	12	12	4	7	4	4	4	12	5	12
	cl	250	250	-	-	117	217	-	-	133	217	-	-
$\Delta t = 3$	κ	25	25	25	22	25	25	25	20	25	23	25	22
	p	8	9	9	9	4	4	4	4	9	9	4	4
	cl	200	233	-	-	200	183	-	-	183	200	-	-
$\Delta t = 4$	κ	25	25	25	22	25	25	25	22	25	25	25	18
	p	8	8	8	8	4	5	4	4	8	12	4	5
	cl	217	183	-	-	183	200	-	-	167	233	-	-

of this wind farm and the 1% ramps mean $\Delta p_{\mathrm{th}} \simeq 30\%$, a value similar to those used in other studies [Bossavy et al., 2013a; Bradford et al., 2010].

Parameter selection. Performance will strongly depend on the concrete selection of the parameters used, namely the number κ of patterns closest to $\mathbf{x}^{(\tau)}$, the number p of delays included in the patterns, and the value of cl used to determine the covariance cloud.

These parameters have been chosen using a validation set and selecting the best parameters from a grid with nodes defined by $\kappa = \{15, 20, 25\}$, $p = \{4, 8, 12\}$ and $cl = \{100, 150, 200, 250\}$, with the optimality criterion being the best AUC-ROC obtained with each model.

Optimal parameters for each model (i.e., the one with Euclidean distance ($\mathcal{NN}^{\mathrm{E}}$) or the one with Mahalanobis distance ($\mathcal{NN}^{\mathrm{M_T}}$)) according to the pattern information that they contain (i.e., without prediction, with daily predictions or with hourly predictions) are shown in Table 4.6. To select these final parameters, we have averaged and rounded the optimal values obtained for each different horizon used ($H = \{0, 1, 2\}$).

The ridge penalty for the RLS model has also been chosen using the second year for validation and applying the model throughout the third year. For example for the daily model, a $\gamma = 9.54e+03$ value has been selected, resulting in a test error of 8.987%. For the hourly model, the ridge penalty has been selected separately for each horizon, obtaining accordingly a different test error value.

Results obtained. The two κ-nearest neighbors models have been applied to this wind farm data using the parameters specified above, and they have been compared against the baseline models.

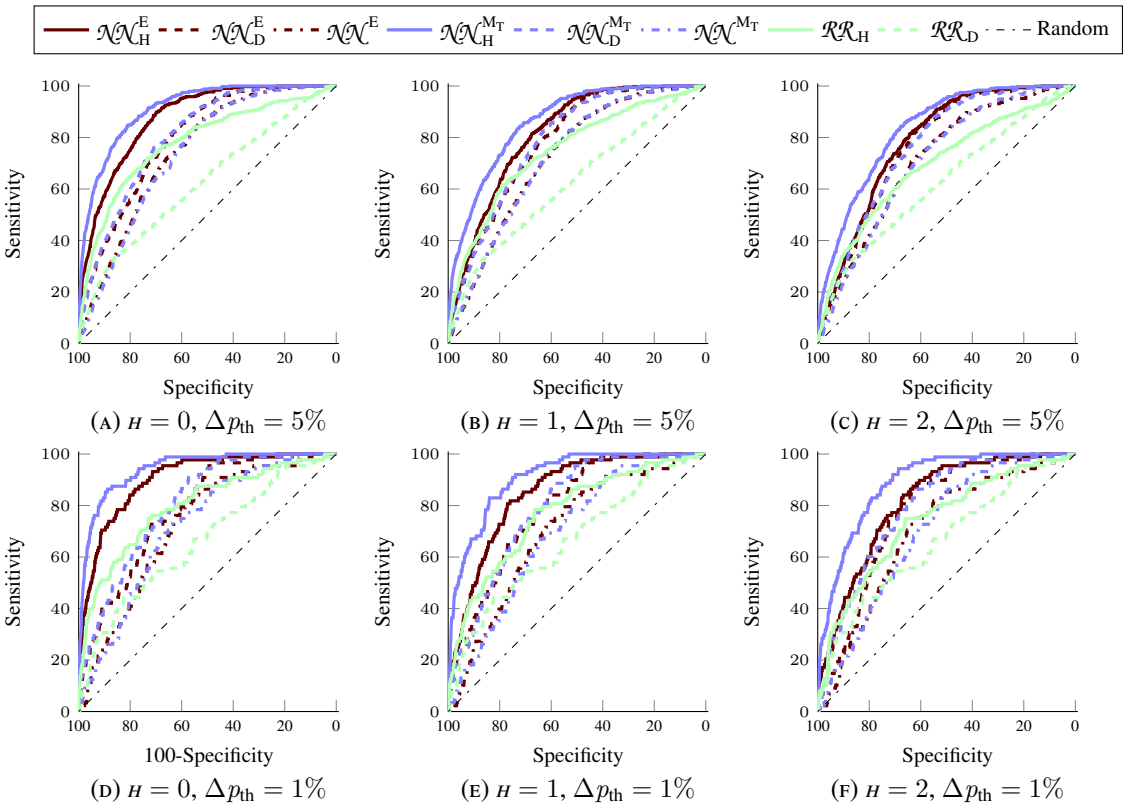
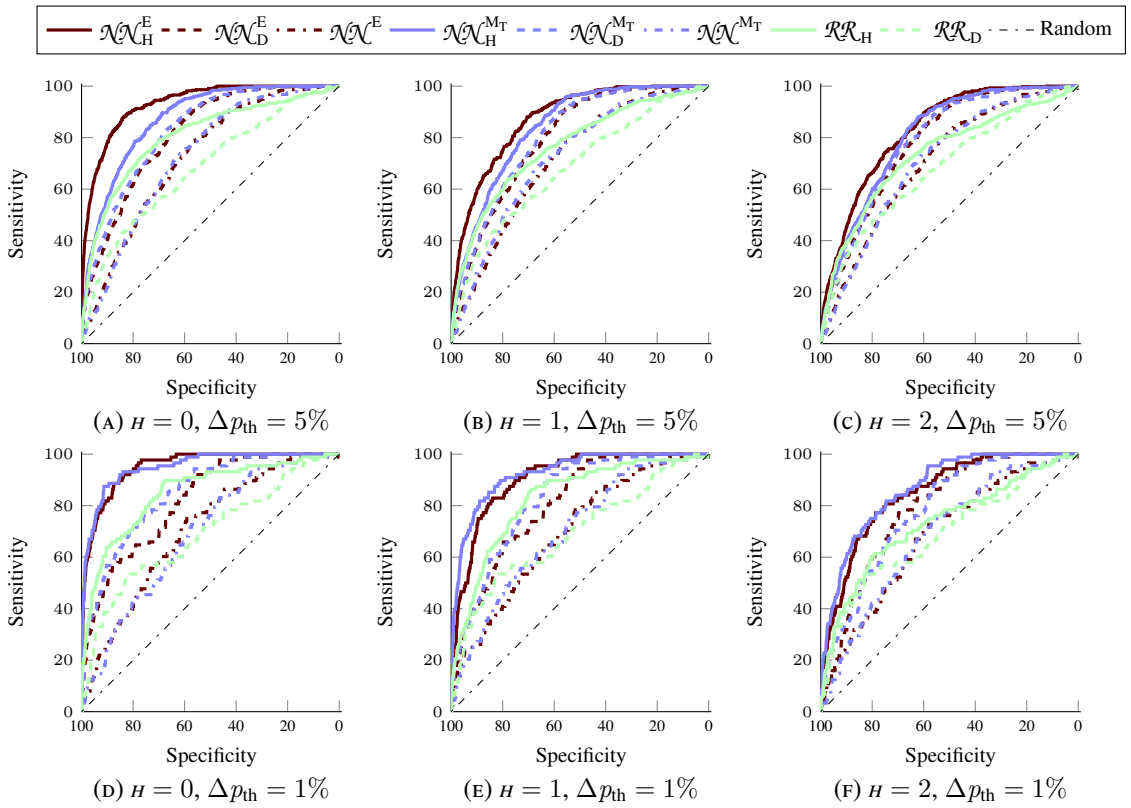


FIGURE 4.7: Sotavento ROC $\Delta t = 2$ curves for $H = \{0, 1, 2\}$ and $\Delta p_{th} = \{1\%, 5\%\}$.

Figure 4.7, Figure 4.8 and Figure 4.9 represent the ROC curves for each model for every horizon and power threshold considered. Each image shows the results for each optimal model, and on them random models have been depicted in black, RLS models are represented in green lines, Euclidean models are depicted in dark red and the anisotropic distance based models are pictured with blue lines. Different symbols have been also used to differentiate the pattern information used to build each model, with the dash-point combination being used for the plain pattern models, dash lines for daily pattern models, and solid lines for the hourly pattern models.

It can be observed how models appear in blocks according to the pattern information they are built with, being worse those models without predictions and being the best ones those with hourly predictions, as it was to be expected. Among these leader curves, the winner for all the horizons when $\Delta t = 2$ seems to be the Mahalanobis distance based model reinforced with hourly predictions. This is the most interesting case, as it presents big increases of energy in a very small time interval. For $\Delta t = 3$ Euclidean distance based model seem to be better and for $\Delta t = 4$ Mahalanobis and Euclidean distance based models reinforced with hourly predictions seem to present equal results.

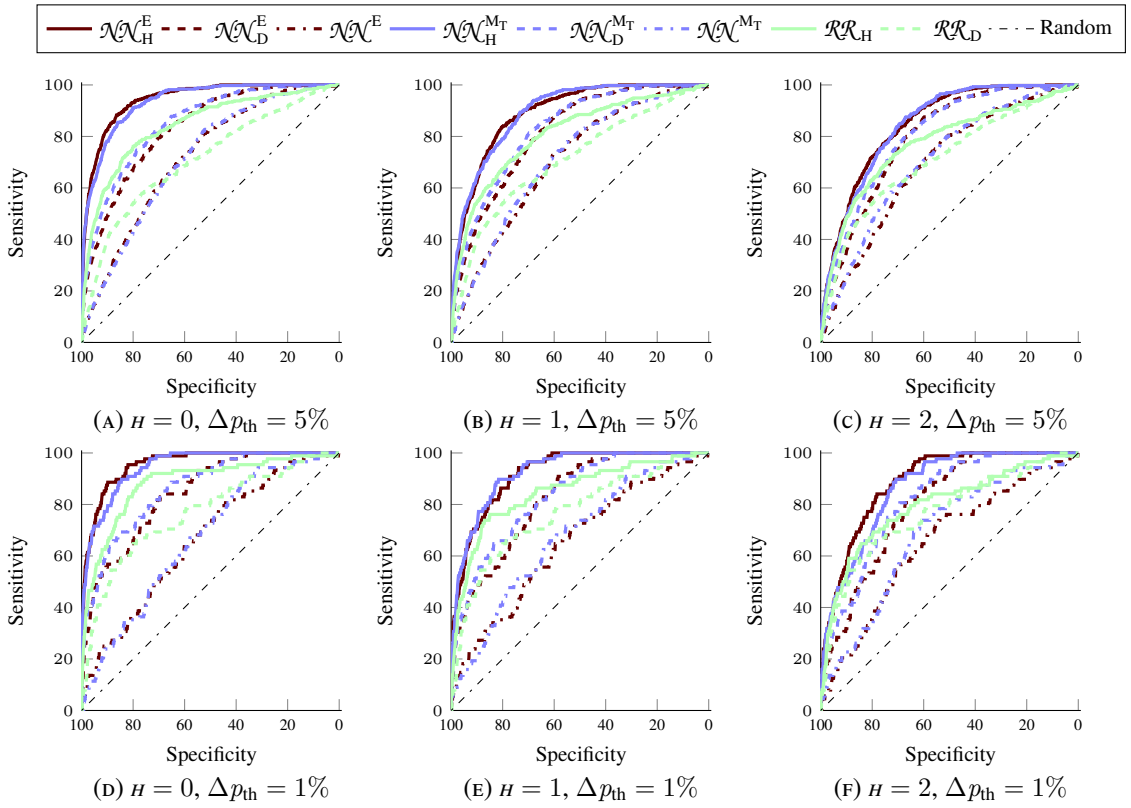
Figure 4.10, Figure 4.11 and Figure 4.12 show PR curves for each Δt proven. These curves are a good visual representation of classification problems when there exists a large skew in the class distribution, as it is our case, and in the picture we can appreciated how the same behavior shown



(A) $H = 0$, $\Delta p_{\text{th}} = 5\%$ (B) $H = 1$, $\Delta p_{\text{th}} = 5\%$ (C) $H = 2$, $\Delta p_{\text{th}} = 5\%$

(D) $H = 0$, $\Delta p_{\text{th}} = 1\%$ (E) $H = 1$, $\Delta p_{\text{th}} = 1\%$ (F) $H = 2$, $\Delta p_{\text{th}} = 1\%$

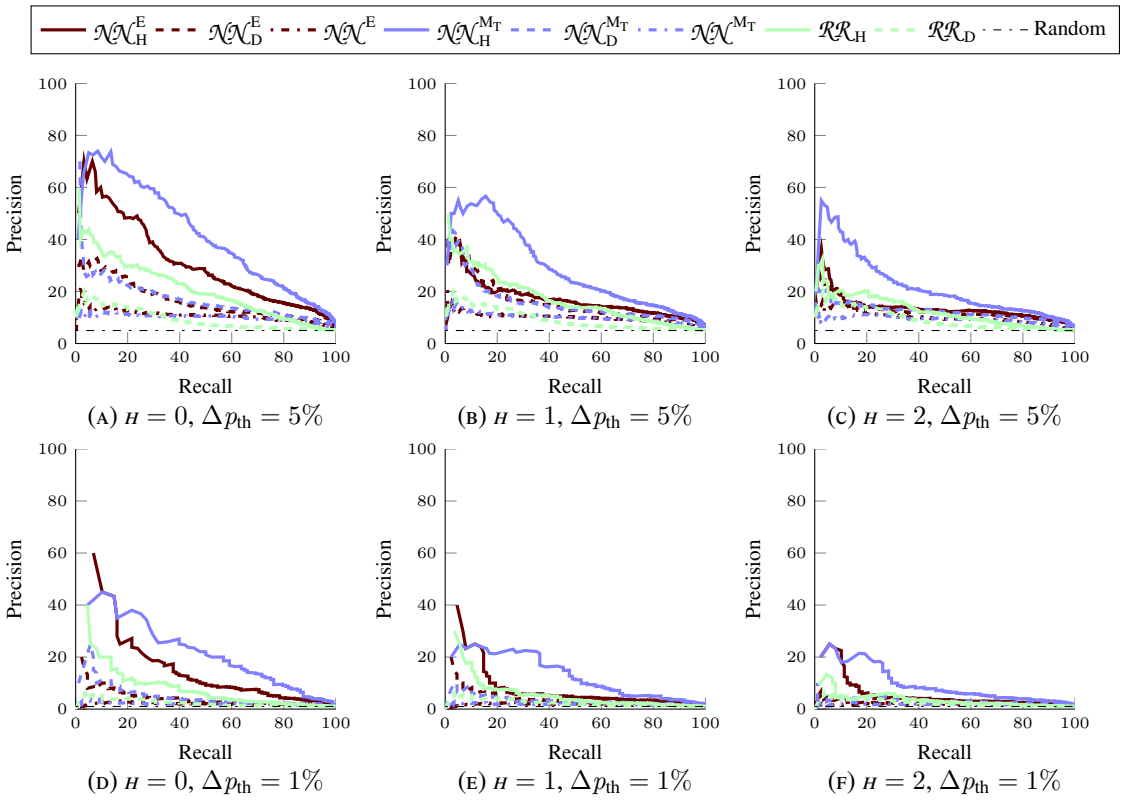
FIGURE 4.8: Sotavento ROC $\Delta t = 3$ curves for $H = \{0, 1, 2\}$ and $\Delta p_{\text{th}} = \{1\%, 5\%\}$.



(A) $H = 0$, $\Delta p_{\text{th}} = 5\%$ (B) $H = 1$, $\Delta p_{\text{th}} = 5\%$ (C) $H = 2$, $\Delta p_{\text{th}} = 5\%$

(D) $H = 0$, $\Delta p_{\text{th}} = 1\%$ (E) $H = 1$, $\Delta p_{\text{th}} = 1\%$ (F) $H = 2$, $\Delta p_{\text{th}} = 1\%$

FIGURE 4.9: Sotavento ROC $\Delta t = 4$ curves for $H = \{0, 1, 2\}$ and $\Delta p_{\text{th}} = \{1\%, 5\%\}$.



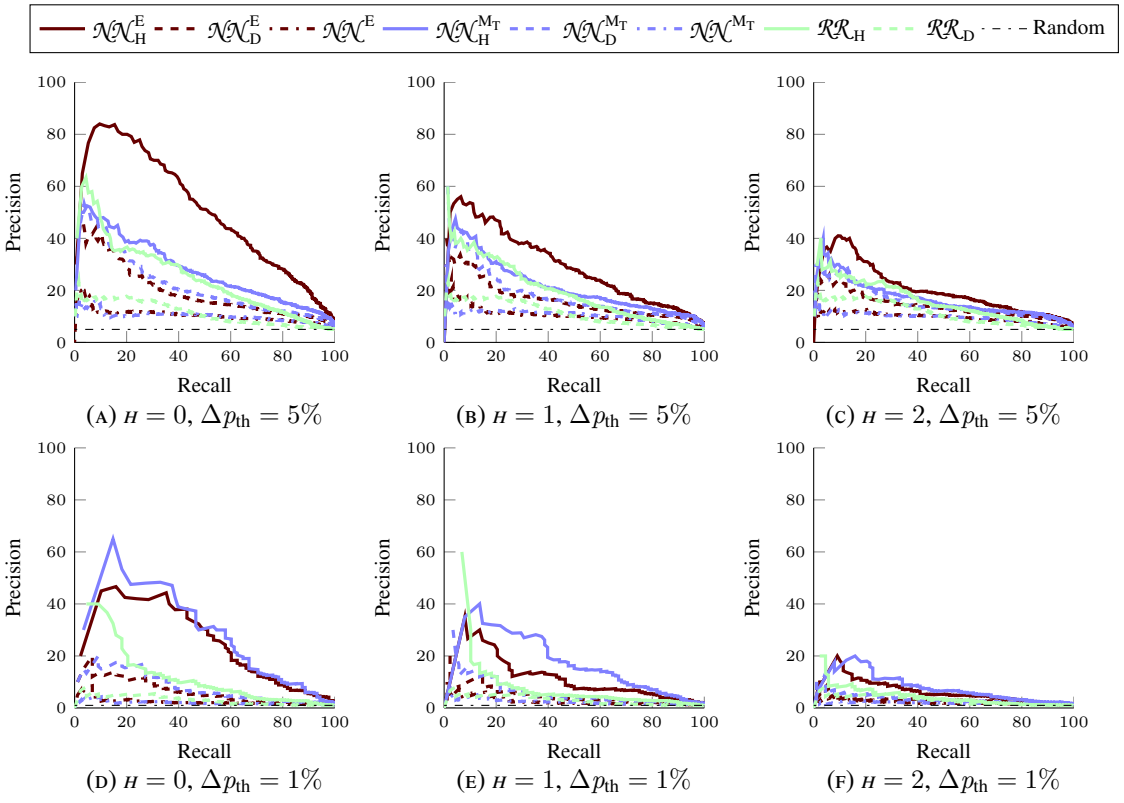
(A) $H = 0$, $\Delta p_{\text{th}} = 5\%$ (B) $H = 1$, $\Delta p_{\text{th}} = 5\%$ (C) $H = 2$, $\Delta p_{\text{th}} = 5\%$

(D) $H = 0$, $\Delta p_{\text{th}} = 1\%$ (E) $H = 1$, $\Delta p_{\text{th}} = 1\%$ (F) $H = 2$, $\Delta p_{\text{th}} = 1\%$

FIGURE 4.10: Sotavento PR $\Delta t = 2$ curves for $H = \{0, 1, 2\}$ and $\Delta p_{\text{th}} = \{1\%, 5\%\}$.

in ROC curves is repeated here but the advantage of the winning models is more clearly seen in this kind of curves. They also give valuable information to estimate practical aspects of model's performance, as they measure the number of ramps detected (recall) against the percentage of correct alarms emitted (precision). For example, to catch the 40% of actual ramps in the $H = 0$, $\Delta p_{\text{th}} = 5\%, \Delta t = 2$ scenario, the PR curves show that we will generate a 40% of false alarms when we work with $\mathcal{NN}^{M_T}_{H}$ but about 70% of false alarms if $\mathcal{NN}^{E}_{H}$ is used.

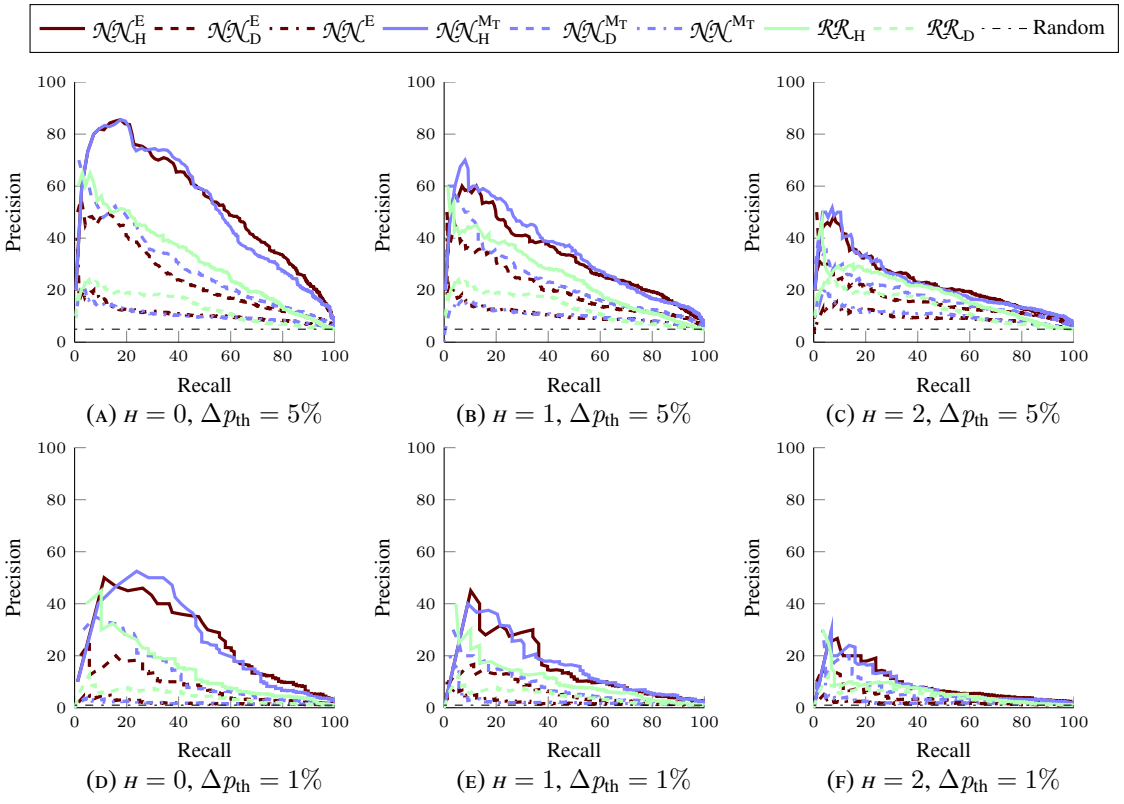
To reinforce these results, we present in Table 4.7, Table 4.8 and Table 4.9 the corresponding AUC values for every model. To determine whether a difference between AUCs is statistically significant for ROC curves we have used the DeLong's test [DeLong et al., 1988], which is based on the Mann–Whitney statistic. For this purpose we have used the R routine provided by Robin et al. [2011]. The largest AUC values are shown in boldface when the DeLong's test shows them to be significantly higher than the next AUC values. When more than one value has been highlighted, we can consider those models as having the same performance, as there is no statistically evidence to state they are different. Although DeLong's test cannot be applied for the AUC of the PR curves, we still give its values for these curves for exploratory purposes; accordingly, here we have not highlighted any value as we cannot derive any statistical significance from them. These measures confirm the previous conclusions obtained by visually analyzing the ROC and PR curves.



(A) $H = 0$, $\Delta p_{\text{th}} = 5\%$ (B) $H = 1$, $\Delta p_{\text{th}} = 5\%$ (C) $H = 2$, $\Delta p_{\text{th}} = 5\%$

(D) $H = 0$, $\Delta p_{\text{th}} = 1\%$ (E) $H = 1$, $\Delta p_{\text{th}} = 1\%$ (F) $H = 2$, $\Delta p_{\text{th}} = 1\%$

FIGURE 4.11: Sotavento PR $\Delta t = 3$ curves for $H = \{0, 1, 2\}$ and $\Delta p_{\text{th}} = \{1\%, 5\%\}$.



(A) $H = 0$, $\Delta p_{\text{th}} = 5\%$ (B) $H = 1$, $\Delta p_{\text{th}} = 5\%$ (C) $H = 2$, $\Delta p_{\text{th}} = 5\%$

(D) $H = 0$, $\Delta p_{\text{th}} = 1\%$ (E) $H = 1$, $\Delta p_{\text{th}} = 1\%$ (F) $H = 2$, $\Delta p_{\text{th}} = 1\%$

FIGURE 4.12: Sotavento PR $\Delta t = 4$ curves for $H = \{0, 1, 2\}$ and $\Delta p_{\text{th}} = \{1\%, 5\%\}$.

TABLE 4.7: AUC-ROC and AUC-PR for the different models and settings with $\Delta t = 2$.

	Mod	$\mathcal{NN}_{\mathrm{H}}^{\mathrm{E}}$	$\mathcal{NN}_{\mathrm{H}}^{\mathrm{M_T}}$	$\mathcal{RR}_{\mathrm{H}}$	$\mathcal{NN}_{\mathrm{D}}^{\mathrm{E}}$	$\mathcal{NN}_{\mathrm{D}}^{\mathrm{M_T}}$	$\mathcal{RR}_{\mathrm{D}}$	$\mathcal{NN}^{\mathrm{E}}$	$\mathcal{NN}^{\mathrm{M_T}}$
$\rho = 5\%$					ROC				
	$\mathcal{H} = 0$	0.876	**0.911**	0.780	0.796	0.805	0.624	0.745	0.736
	$\mathcal{H} = 1$	0.815	**0.862**	0.750	0.785	0.788	0.624	0.723	0.724
	$\mathcal{H} = 2$	0.786	**0.831**	0.692	0.770	0.767	0.624	0.712	0.708
					PR				
	$\mathcal{H} = 0$	0.308	0.416	0.209	0.166	0.173	0.092	0.109	0.100
	$\mathcal{H} = 1$	0.172	0.287	0.171	0.161	0.156	0.092	0.100	0.100
	$\mathcal{H} = 2$	0.136	0.218	0.125	0.124	0.124	0.092	0.101	0.093
$\rho = 1\%$					ROC				
	$\mathcal{H} = 0$	0.904	**0.942**	0.798	0.787	0.804	0.660	0.723	0.719
	$\mathcal{H} = 1$	0.850	**0.909**	0.771	0.790	0.792	0.660	0.699	0.694
	$\mathcal{H} = 2$	0.815	**0.888**	0.741	0.777	0.795	0.660	0.698	0.701
					PR				
	$\mathcal{H} = 0$	0.162	0.219	0.086	0.045	0.051	0.024	0.020	0.021
	$\mathcal{H} = 1$	0.077	0.132	0.063	0.040	0.035	0.024	0.018	0.018
	$\mathcal{H} = 2$	0.058	0.095	0.036	0.026	0.031	0.024	0.018	0.018

TABLE 4.8: AUC-ROC and AUC-PR for the different models and settings with $\Delta t = 3$.

	Mod	$\mathcal{NN}_{\mathrm{H}}^{\mathrm{E}}$	$\mathcal{NN}_{\mathrm{H}}^{\mathrm{M_T}}$	$\mathcal{RR}_{\mathrm{H}}$	$\mathcal{NN}_{\mathrm{D}}^{\mathrm{E}}$	$\mathcal{NN}_{\mathrm{D}}^{\mathrm{M_T}}$	$\mathcal{RR}_{\mathrm{D}}$	$\mathcal{NN}^{\mathrm{E}}$	$\mathcal{NN}^{\mathrm{M_T}}$
$\rho = 5\%$					ROC				
	$\mathcal{H} = 0$	**0.931**	0.873	0.807	0.813	0.827	0.682	0.723	0.726
	$\mathcal{H} = 1$	**0.872**	0.840	0.766	0.798	0.807	0.682	0.715	0.723
	$\mathcal{H} = 2$	**0.829**	0.807	0.735	0.782	0.787	0.682	0.705	0.708
					PR				
	$\mathcal{H} = 0$	0.504	0.275	0.251	0.195	0.219	0.112	0.103	0.101
	$\mathcal{H} = 1$	0.293	0.214	0.196	0.163	0.186	0.112	0.106	0.104
	$\mathcal{H} = 2$	0.202	0.167	0.156	0.139	0.152	0.112	0.096	0.097
$\rho = 1\%$					ROC				
	$\mathcal{H} = 0$	**0.952**	**0.950**	0.851	0.819	0.850	0.691	0.705	0.688
	$\mathcal{H} = 1$	0.907	**0.927**	0.829	0.807	0.827	0.691	0.695	0.699
	$\mathcal{H} = 2$	**0.849**	**0.868**	0.731	0.789	0.782	0.691	0.683	0.687
					PR				
	$\mathcal{H} = 0$	0.258	0.297	0.126	0.063	0.081	0.030	0.024	0.020
	$\mathcal{H} = 1$	0.119	0.181	0.091	0.053	0.059	0.030	0.027	0.024
	$\mathcal{H} = 2$	0.065	0.080	0.044	0.030	0.035	0.030	0.021	0.019

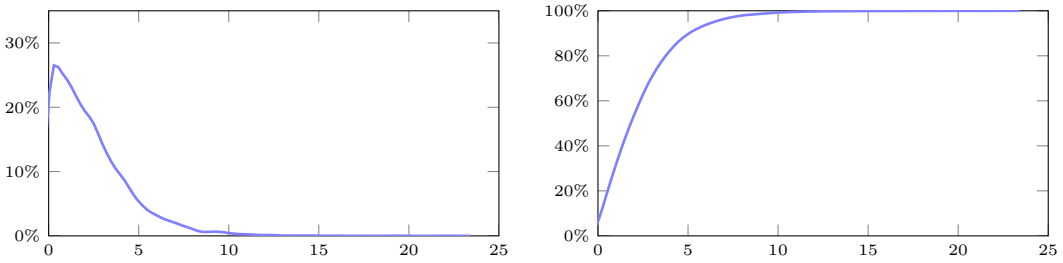
TABLE 4.9: AUC-ROC and AUC-PR for the different models and settings with $\Delta t = 4$.

	Mod	$\mathcal{NN}_{\mathrm{H}}^{\mathrm{E}}$	$\mathcal{NN}_{\mathrm{H}}^{\mathrm{M_T}}$	$\mathcal{RR}_{\mathrm{H}}$	$\mathcal{NN}_{\mathrm{D}}^{\mathrm{E}}$	$\mathcal{NN}_{\mathrm{D}}^{\mathrm{M_T}}$	$\mathcal{RR}_{\mathrm{D}}$	$\mathcal{NN}^{\mathrm{E}}$	$\mathcal{NN}^{\mathrm{M_T}}$
$\rho = 5\%$					ROC				
	$H = 0$	**0.943**	**0.934**	0.844	0.831	0.851	0.710	0.718	0.714
	$H = 1$	**0.893**	**0.895**	0.809	0.816	0.829	0.710	0.706	0.710
	$H = 2$	**0.848**	**0.847**	0.771	0.796	0.806	0.710	0.692	0.707
					PR				
	$H = 0$	0.536	0.520	0.314	0.247	0.288	0.133	0.110	0.104
	$H = 1$	0.327	0.348	0.249	0.207	0.238	0.133	0.108	0.101
	$H = 2$	0.229	0.229	0.186	0.158	0.175	0.133	0.094	0.104
$\rho = 1\%$					ROC				
	$H = 0$	**0.956**	**0.946**	0.884	0.844	0.863	0.766	0.662	0.667
	$H = 1$	**0.920**	**0.926**	0.849	0.823	0.834	0.766	0.651	0.673
	$H = 2$	**0.886**	**0.866**	0.795	0.799	0.817	0.766	0.652	0.683
					PR				
	$H = 0$	0.268	0.267	0.154	0.090	0.136	0.044	0.020	0.018
	$H = 1$	0.161	0.166	0.109	0.062	0.082	0.044	0.023	0.035
	$H = 2$	0.093	0.088	0.066	0.043	0.059	0.044	0.018	0.031

4.5.3.3 Global Wind Ramp Prediction for Spain

Problem to solve. The previous experiments have been repeated with global wind energy data production for Spain provided by Red Eléctrica de España (REE), Spanish Transmission System Operator (TSO). In particular, the data set used is composed of hourly productions from January 1, 2011, to December 31, 2013. The first year will be used as the train set, the second year will be considered for validation purposes and the last year of the data set for testing. For the ramp prediction model, the set of past data over which we will look for neighbors is again formed by one year up to the hour before the current hour for which ramps are to be predicted. Now, all the wind power productions are given as percentages of Spain's wind power installed capacity.

For these global energy data, if we consider $\Delta t = 2$, a 5% ramps mean a $\Delta p_{\text{th}} \simeq 7\%$ of the total wind power in Spain which is a relatively low percentage, due to the smoothness effect obtained when the whole country production is considered. Thus, in the experiments here we would like to evaluate ramps with a higher impact and we will consider 1% ramps, which means a $\Delta p_{\text{th}} \simeq 10\%$, and 0.5% ramps related to power increments higher than a 11% of the total wind power in Spain. Figure 4.13 shows the distribution of the $\Delta t = 2$ absolute differences of wind energy production in Spain. In the right hand image the cumulative density function is depicted, where we can observe that the curve becomes almost constant from an increase of 10%, which corresponds to a 99% increments percentile. This illustrates the small difference between the 99% and 99.5% percentiles. On the left hand side image the probability density function is plotted, and we can see that an increase higher than the 11% of the total wind power in Spain is an very extreme enough event.



(A) Probability Density Function (B) Cumulative Density Function

FIGURE 4.13: Distribution of the wind energy $\Delta t = 2$ absolute differences for Spanish productions.

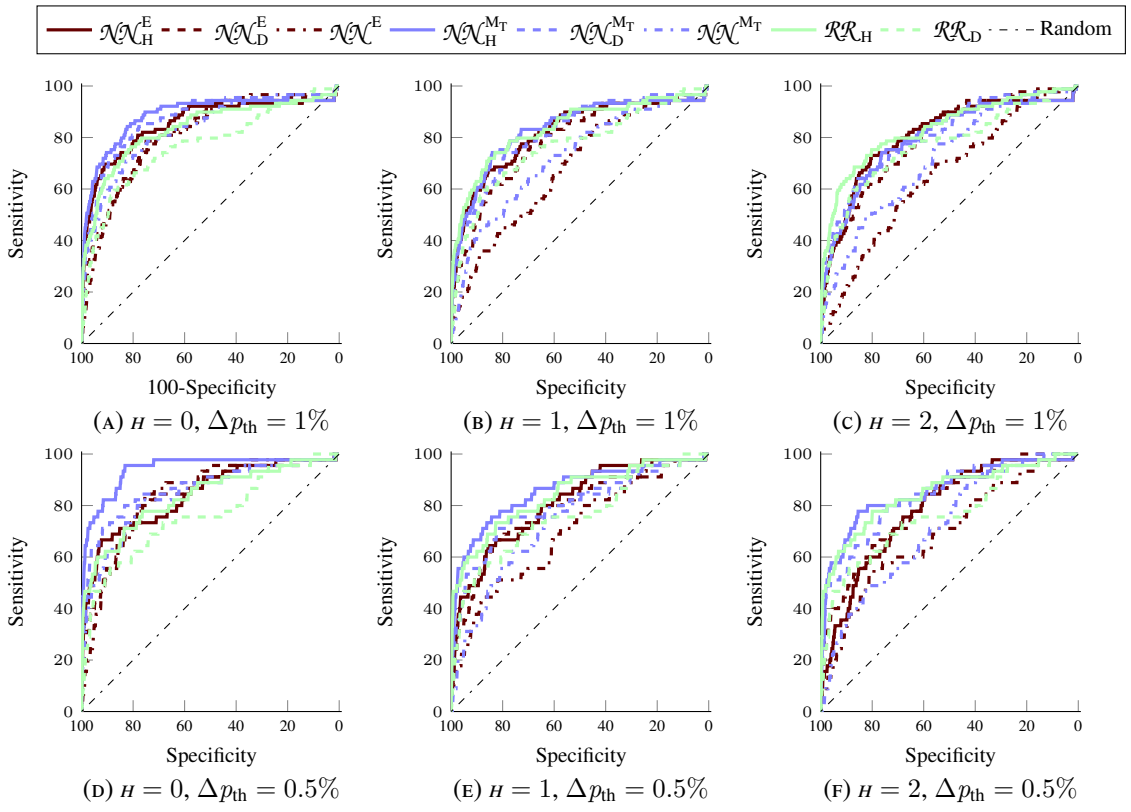
TABLE 4.10: Best parameter values of κ, p and cL for each model.

	Parameter	$\mathcal{NN}_{\mathrm{H}}^{\mathrm{M_T}}$		$\mathcal{NN}_{\mathrm{H}}^{\mathrm{E}}$		$\mathcal{NN}_{\mathrm{D}}^{\mathrm{M_T}}$		$\mathcal{NN}_{\mathrm{D}}^{\mathrm{E}}$		$\mathcal{NN}^{\mathrm{M_T}}$		$\mathcal{NN}^{\mathrm{E}}$	
		1%	0.5%	1%	0.5%	1%	0.5%	1%	0.5%	1%	0.5%	1%	0.5%
$\Delta t = 2$	κ	22	20	22	18	22	22	25	20	25	25	18	18
	p	12	11	11	9	4	5	4	4	9	7	4	4
	cL	250	167	-	-	217	233	-	-	233	183	-	-
$\Delta t = 3$	κ	25	22	25	23	23	22	17	20	23	23	15	18
	p	8	9	11	8	7	9	4	4	7	11	5	5
	cL	250	250	-	-	233	117	-	-	217	183	-	-
$\Delta t = 4$	κ	25	23	20	20	23	23	15	15	25	25	15	15
	p	8	8	8	8	5	5	4	4	9	12	5	7
	cL	233	200	-	-	217	200	-	-	200	200	-	-

Parameter selection. The best parameter values for κ, p and cL have been obtaining doing an exhaustive search over a grid with nodes defined by $\kappa = \{15, 20, 25\}$, $p = \{4, 8, 12\}$ and $cL = \{100, 150, 200, 250\}$. As in the previous example, the selection has been made according to the best AUC-ROC obtained with each model over the validation data set and, as before, to obtain the final parameters, we have averaged and rounded the optimal values obtained for each different horizon used ($H = \{0, 1, 2\}$).

The values obtained for each model (i.e., the Euclidean distance model ($\mathcal{NN}^{\mathrm{E}}$) or Mahalanobis distance model ($\mathcal{NN}^{\mathrm{M_T}}$)) according to the pattern information that they contain (without prediction, with daily predictions or with hourly predictions) are shown in Table 4.10.

A value of $\gamma = 9.54e{+}03$ has been fixed for the penalty factor in the daily RLS model, obtaining a test error of 2.962%. For the hourly model, the ridge penalty has been also set separately for each horizon, obtaining accordingly different test error values.



(A) $H = 0$, $\Delta p_{\text{th}} = 1\%$ (B) $H = 1$, $\Delta p_{\text{th}} = 1\%$ (C) $H = 2$, $\Delta p_{\text{th}} = 1\%$

(D) $H = 0$, $\Delta p_{\text{th}} = 0.5\%$ (E) $H = 1$, $\Delta p_{\text{th}} = 0.5\%$ (F) $H = 2$, $\Delta p_{\text{th}} = 0.5\%$

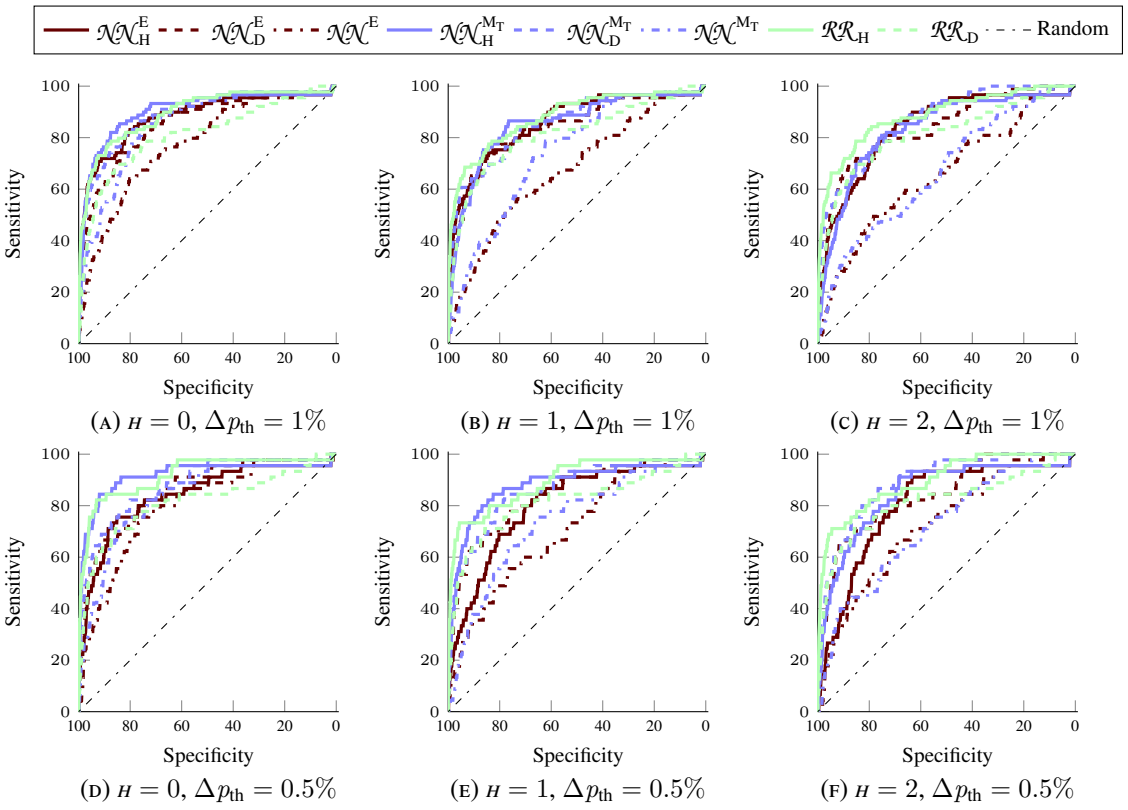
FIGURE 4.14: Spanish ROC $\Delta t = 2$ curves for $H = \{0, 1, 2\}$ and $\Delta p_{\text{th}} = \{0.5\%, 1\%\}$.

Results obtained. The two κ-nearest neighbors models have been applied to these data using the parameters specified above, and they have been also compared against the baseline models.

Figure 4.14, Figure 4.15 and Figure 4.16 represent the ROC curves for each model for the three horizons considered ($H = \{0, 1, 2\}$) and the two power thresholds fixed ($\Delta p_{\text{th}} = \{0.5, 1\}\%$), using the same color code explained for the Sotavento's example.

In these images, we can appreciate first a stairway-effect, due to the low percentages we are taken into account in these experiments that reduced considerably the number of real positive ramps. On the other side, we observe that the curves of the different models appear closer than those in Sotavento's case. For horizon $H = 0$, $\mathcal{NN}_{\mathrm{H}}^{\mathrm{M_T}}$ seems to be the winner for all the Δp_{th} and Δt considered, except for $\Delta t = 4$ where its advantage is less clear. For the other horizons, we cannot visually determine a clear winner but it can be said that that the $\mathcal{NN}_{\mathrm{H}}^{\mathrm{M_T}}$ model always appears at the top in every image, matched closely by the $\mathcal{RR}_{\mathrm{H}}$ one. This RLS model performs here much better than for Sotavento's wind farm, the reason possibly being that the smoother evolution of global wind power results in a much better numerical prediction by the $\mathcal{RR}_{\mathrm{H}}$ model than that achievable in Sotavento.

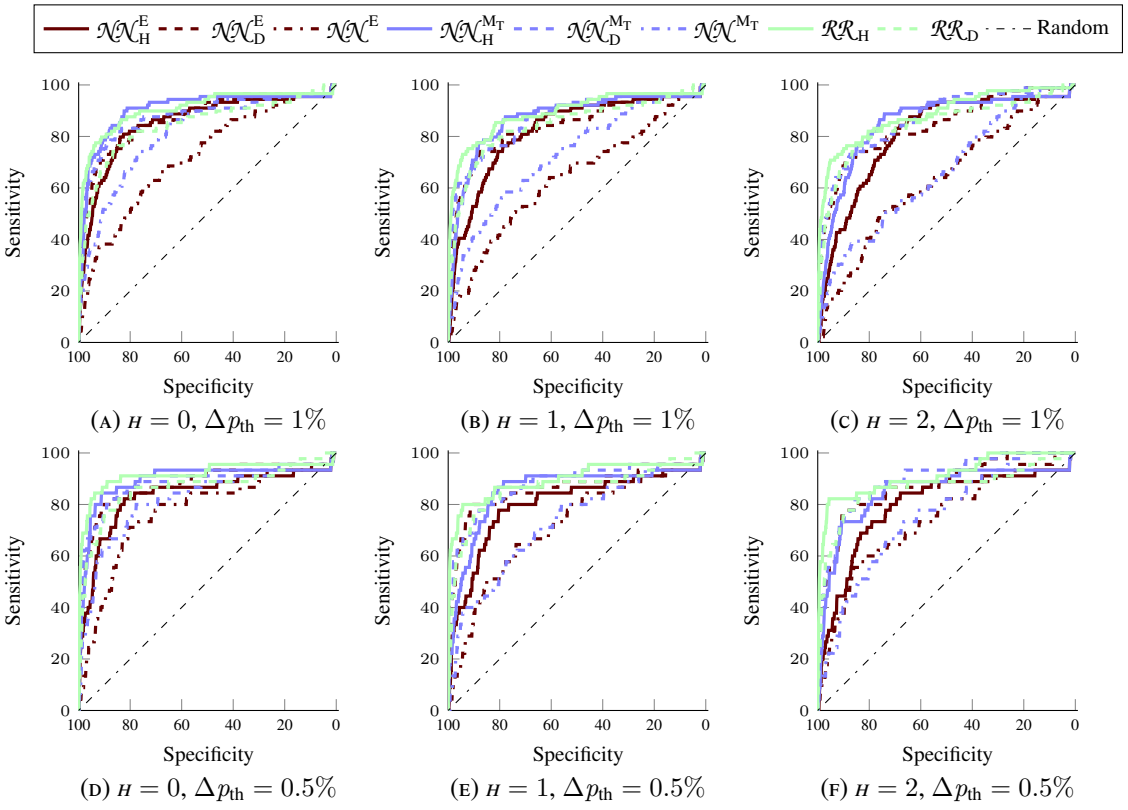
Figure 4.17, Figure 4.18 and Figure 4.19 show PR curves for each Δt considered. Notice that ramp detection is a very unbalanced problem, particularly in the $\Delta p_{\text{th}} = 0.5\%$ case, which is



(A) $H = 0$, $\Delta p_{\text{th}} = 1\%$ (B) $H = 1$, $\Delta p_{\text{th}} = 1\%$ (C) $H = 2$, $\Delta p_{\text{th}} = 1\%$

(D) $H = 0$, $\Delta p_{\text{th}} = 0.5\%$ (E) $H = 1$, $\Delta p_{\text{th}} = 0.5\%$ (F) $H = 2$, $\Delta p_{\text{th}} = 0.5\%$

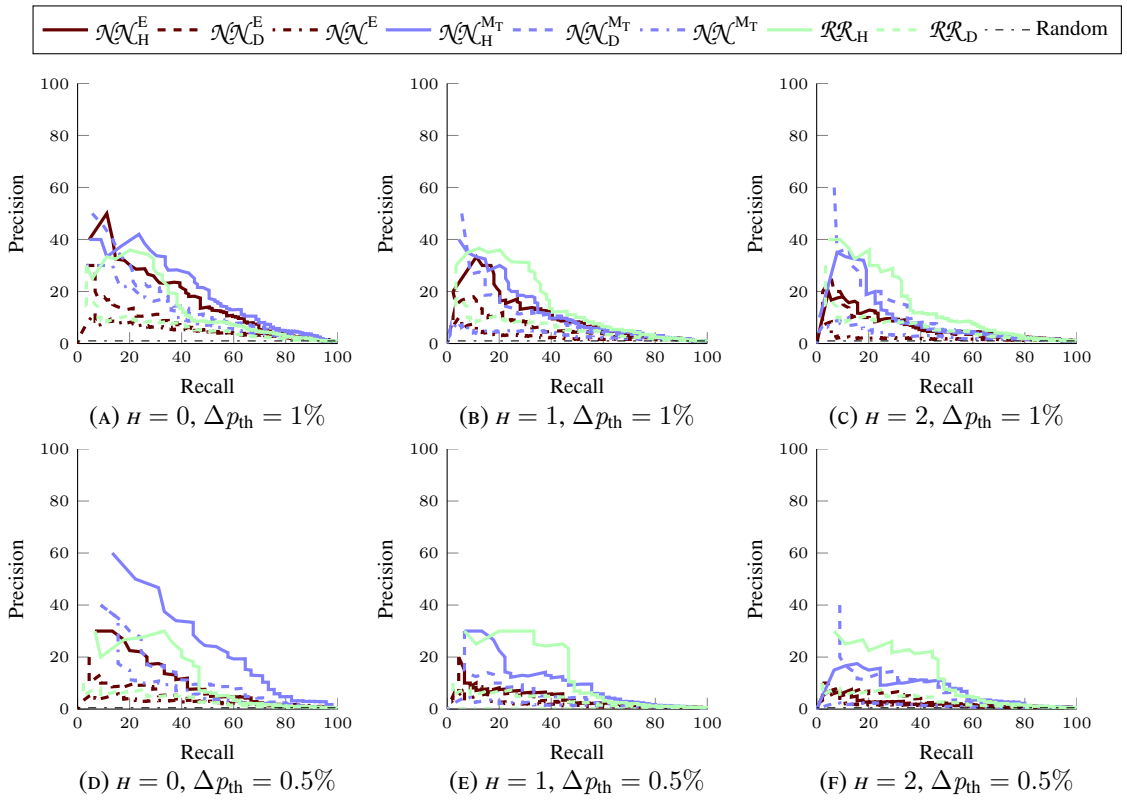
FIGURE 4.15: Spanish ROC $\Delta t = 3$ curves for $H = \{0, 1, 2\}$ and $\Delta p_{\text{th}} = \{0.5\%, 1\%\}$.



(A) $H = 0$, $\Delta p_{\text{th}} = 1\%$ (B) $H = 1$, $\Delta p_{\text{th}} = 1\%$ (C) $H = 2$, $\Delta p_{\text{th}} = 1\%$

(D) $H = 0$, $\Delta p_{\text{th}} = 0.5\%$ (E) $H = 1$, $\Delta p_{\text{th}} = 0.5\%$ (F) $H = 2$, $\Delta p_{\text{th}} = 0.5\%$

FIGURE 4.16: Spanish ROC $\Delta t = 4$ curves for $H = \{0, 1, 2\}$ and $\Delta p_{\text{th}} = \{0.5\%, 1\%\}$.



(A) $H = 0$, $\Delta p_{\text{th}} = 1\%$ (B) $H = 1$, $\Delta p_{\text{th}} = 1\%$ (C) $H = 2$, $\Delta p_{\text{th}} = 1\%$

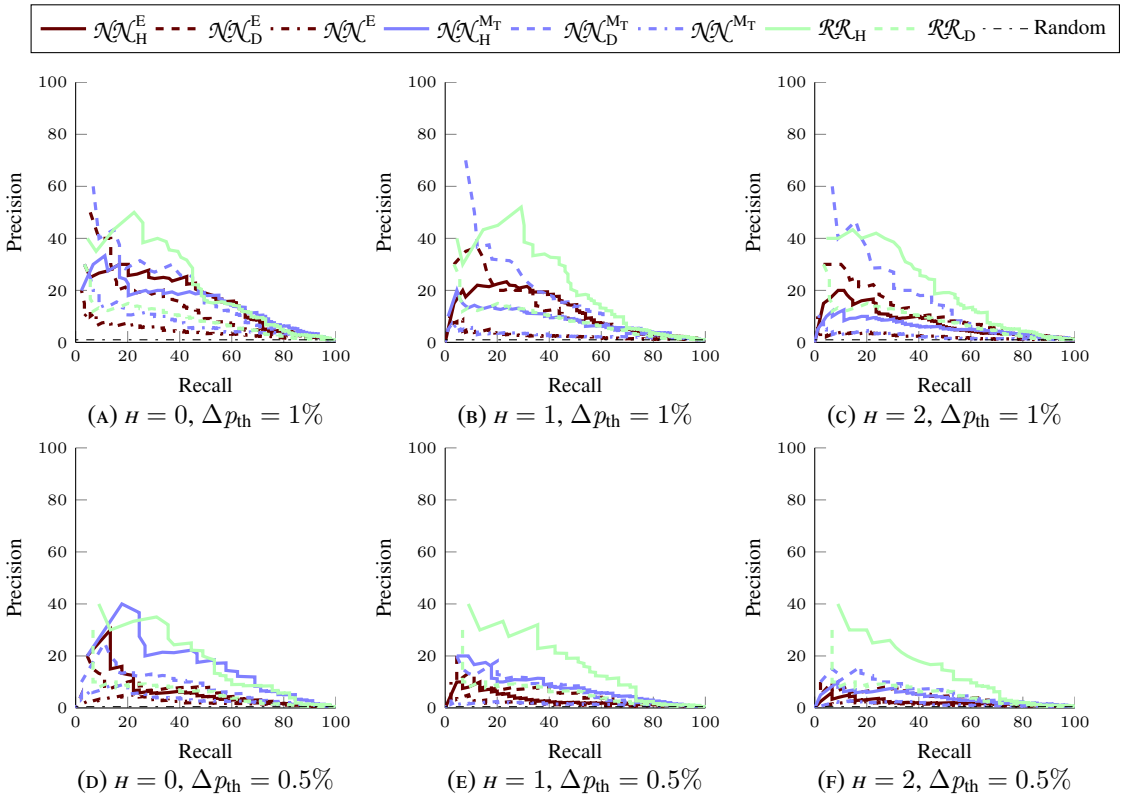
(D) $H = 0$, $\Delta p_{\text{th}} = 0.5\%$ (E) $H = 1$, $\Delta p_{\text{th}} = 0.5\%$ (F) $H = 2$, $\Delta p_{\text{th}} = 0.5\%$

FIGURE 4.17: Spanish PR $\Delta t = 2$ curves for $H = \{0, 1, 2\}$ and $\Delta p_{\text{th}} = \{0.5\%, 1\%\}$.

thus a difficult classification problem. The PR curves let us again analyze the balance between the percentage of ramps that we would like to catch versus the percentage of false alarms that we are willing to assume. In this case, we can appreciate that considering for instance the horizon $H = 0$, $\Delta t = 2$ and $\Delta p_{\text{th}} = 1\%$ case, to detect a 40% of the ramps we will have around a 70% of false alarms when using the best model, namely $\mathcal{NN}^{M_T}_{H}$. If we want, for example, to recover a more demanding 60% of the ramps, more than the 80% of the alarms generated will be false.

Again the AUC values will be the selected metric for evaluating each model; the values obtained are shown in Table 4.11, Table 4.12 and Table 4.13. To determine whether a difference between AUCs is statistically significant we have applied again the DeLong's test, highlighting in boldface the largest values and the ties occurred. Recall that this test cannot be applied to the AUC values of the PR curves and is given just for exploratory purposes; accordingly, here we have not highlighted higher values.

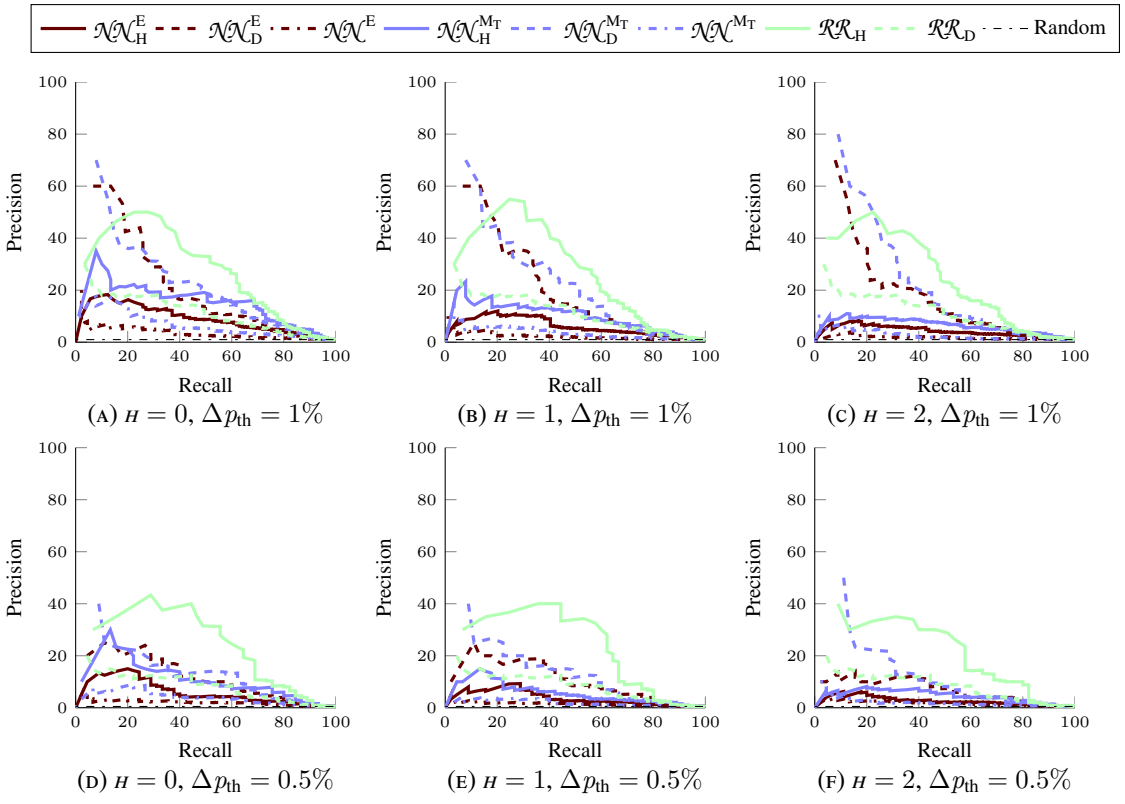
Results show that whereas it is true that models with different pattern information are significantly different, in some cases the NN models are not very different from the RLS ones and in most cases there is not statistical significance to say which is the best model to predict ramps, confirming what we have seen in the ROC and PR plots.



(A) $H = 0$, $\Delta p_{\text{th}} = 1\%$ (B) $H = 1$, $\Delta p_{\text{th}} = 1\%$ (C) $H = 2$, $\Delta p_{\text{th}} = 1\%$

(D) $H = 0$, $\Delta p_{\text{th}} = 0.5\%$ (E) $H = 1$, $\Delta p_{\text{th}} = 0.5\%$ (F) $H = 2$, $\Delta p_{\text{th}} = 0.5\%$

FIGURE 4.18: Spanish PR $\Delta t = 3$ curves for $H = \{0, 1, 2\}$ and $\Delta p_{\text{th}} = \{0.5\%, 1\%\}$.



(A) $H = 0$, $\Delta p_{\text{th}} = 1\%$ (B) $H = 1$, $\Delta p_{\text{th}} = 1\%$ (C) $H = 2$, $\Delta p_{\text{th}} = 1\%$

(D) $H = 0$, $\Delta p_{\text{th}} = 0.5\%$ (E) $H = 1$, $\Delta p_{\text{th}} = 0.5\%$ (F) $H = 2$, $\Delta p_{\text{th}} = 0.5\%$

FIGURE 4.19: Spanish PR $\Delta t = 4$ curves for $H = \{0, 1, 2\}$ and $\Delta p_{\text{th}} = \{0.5\%, 1\%\}$.

TABLE 4.11: AUC-ROC and AUC-PR for the different models and settings with $\Delta t = 2$.

	Mod	$\mathcal{NN}_{\text{H}}^{\text{E}}$	$\mathcal{NN}_{\text{H}}^{\text{M}_\text{T}}$	$\mathcal{RR}_{\text{H}}$	$\mathcal{NN}_{\text{D}}^{\text{E}}$	$\mathcal{NN}_{\text{D}}^{\text{M}_\text{T}}$	$\mathcal{RR}_{\text{D}}$	$\mathcal{NN}^{\text{E}}$	$\mathcal{NN}^{\text{M}_\text{T}}$
					ROC				
$\rho = 0.5\%$	$H = 0$	0.841	**0.941**	0.833	0.837	0.869	0.760	0.834	0.843
	$H = 1$	**0.810**	**0.858**	**0.841**	0.788	0.793	0.760	0.704	0.745
	$H = 2$	0.785	**0.854**	**0.848**	**0.798**	**0.838**	0.760	0.692	0.714
					PR				
	$H = 0$	0.108	0.257	0.130	0.045	0.141	0.034	0.027	0.119
	$H = 1$	0.045	0.101	0.140	0.042	0.083	0.034	0.018	0.017
	$H = 2$	0.023	0.075	0.133	0.039	0.099	0.034	0.016	0.013
					ROC				
$\rho = 1\%$	$H = 0$	0.854	**0.885**	0.834	0.814	**0.871**	0.768	0.809	0.829
	$H = 1$	**0.819**	**0.833**	**0.836**	0.789	**0.819**	0.768	0.684	0.722
	$H = 2$	**0.808**	**0.797**	**0.833**	0.797	**0.807**	0.768	0.650	0.713
					PR				
	$H = 0$	0.174	0.205	0.147	0.085	0.166	0.063	0.047	0.124
	$H = 1$	0.108	0.128	0.155	0.061	0.129	0.063	0.027	0.032
	$H = 2$	0.068	0.100	0.152	0.062	0.128	0.063	0.024	0.032

TABLE 4.12: AUC-ROC and AUC-PR for the different models and settings with $\Delta t = 3$.

	Mod	$\mathcal{NN}_{\text{H}}^{\text{E}}$	$\mathcal{NN}_{\text{H}}^{\text{M}_\text{T}}$	$\mathcal{RR}_{\text{H}}$	$\mathcal{NN}_{\text{D}}^{\text{E}}$	$\mathcal{NN}_{\text{D}}^{\text{M}_\text{T}}$	$\mathcal{RR}_{\text{D}}$	$\mathcal{NN}^{\text{E}}$	$\mathcal{NN}^{\text{M}_\text{T}}$
					ROC				
$\rho = 0.5\%$	$H = 0$	0.841	**0.917**	**0.919**	0.860	0.881	0.814	0.798	0.843
	$H = 1$	0.802	**0.879**	**0.894**	0.841	**0.863**	0.814	0.704	0.740
	$H = 2$	0.806	**0.849**	**0.896**	0.842	**0.892**	0.814	0.724	0.720
					PR				
	$H = 0$	0.067	0.170	0.184	0.062	0.083	0.061	0.022	0.036
	$H = 1$	0.032	0.080	0.182	0.053	0.079	0.061	0.018	0.015
	$H = 2$	0.022	0.041	0.157	0.044	0.067	0.061	0.013	0.014
					ROC				
$\rho = 1\%$	$H = 0$	**0.881**	**0.903**	**0.895**	0.855	**0.891**	0.812	0.775	0.848
	$H = 1$	**0.857**	**0.852**	**0.879**	0.840	0.854	0.812	0.676	0.739
	$H = 2$	**0.841**	0.832	**0.884**	0.839	**0.866**	0.812	0.658	0.668
					PR				
	$H = 0$	0.165	0.157	0.224	0.147	0.220	0.086	0.045	0.083
	$H = 1$	0.118	0.085	0.227	0.137	0.203	0.086	0.026	0.027
	$H = 2$	0.079	0.058	0.211	0.114	0.201	0.086	0.022	0.027

TABLE 4.13: AUC-ROC and AUC-PR for the different models and settings with $\Delta t = 4$.

	Mod	$\mathcal{NN}_{\text{H}}^{\text{E}}$	$\mathcal{NN}_{\text{H}}^{\text{M}_\text{T}}$	$\mathcal{RR}_{\text{H}}$	$\mathcal{NN}_{\text{D}}^{\text{E}}$	$\mathcal{NN}_{\text{D}}^{\text{M}_\text{T}}$	$\mathcal{RR}_{\text{D}}$	$\mathcal{NN}^{\text{E}}$	$\mathcal{NN}^{\text{M}_\text{T}}$
$\rho = 0.5\%$					ROC				
	$\kappa = 0$	0.836	**0.894**	**0.916**	0.880	0.883	0.863	0.770	0.821
	$\kappa = 1$	0.802	0.854	**0.896**	0.863	**0.876**	**0.863**	0.715	0.728
	$\kappa = 2$	0.795	0.851	**0.913**	**0.875**	**0.898**	0.863	0.740	0.751
					PR				
	$\kappa = 0$	0.060	0.113	0.251	0.116	0.140	0.081	0.019	0.038
	$\kappa = 1$	0.037	0.057	0.240	0.104	0.156	0.081	0.014	0.018
	$\kappa = 2$	0.026	0.045	0.211	0.070	0.128	0.081	0.017	0.019
$\rho = 1\%$					ROC				
	$\kappa = 0$	0.854	**0.904**	**0.908**	0.870	0.875	0.844	0.720	0.816
	$\kappa = 1$	0.823	**0.870**	**0.892**	0.843	0.871	0.844	0.639	0.728
	$\kappa = 2$	0.808	**0.853**	**0.885**	0.843	**0.870**	0.844	0.636	0.664
					PR				
	$\kappa = 0$	0.090	0.156	0.277	0.221	0.253	0.112	0.038	0.065
	$\kappa = 1$	0.060	0.100	0.273	0.222	0.262	0.112	0.022	0.037
	$\kappa = 2$	0.043	0.065	0.247	0.184	0.256	0.112	0.018	0.027

4.5.3.4 General Conclusion

Wind ramp detection is a problem of high interest for system operators and wind farm managers and that lends itself to its study under ML methods. In this last section we have proposed a methodology for detecting ramps, considering it as a classification problem that we approach building first patterns made up of previous energy readings and hourly energy predictions derived from NWP forecasts, looking for κ past patterns closest to them under the Anisotropic Diffusion and Euclidean metrics and computing a ramp score from the absolute energy changes in these patterns.

The study of ramps is very recent and there are not reference results with which to compare ours. However, our approach to wind ramps as a classification problem offers a clear way to measure the quality of a wind ramp detection algorithm. The AUC values of ROC and PR curves that we have obtained here show that AD-based ML fits very well with the ramp's temporal structure and yields very good results, clearly the best for the Sotavento wind farm and at the top when Spanish global wind energy is considered. Possible ways of improving the results here are either the consideration of richer patterns than those considered here or the use of more advanced ML algorithms to derive from NWP data more accurately the daily and hourly wind energy forecasts that are the basis of our wind ramp detection approach.

Chapter 5

Out-of-sample Methods

5.1 Introduction

While powerful and elegant, Diffusion Maps (DM) (and, in fact, other manifold learning methods) have a first drawback on the potentially quite costly eigenanalysis of the transition matrix that they require. Observe that in principle sample size coincides with the dimension of the similarity and, hence, transition matrices, which may make its eigenanalysis computationally unfeasible for very large data sets. Although we will not deal with these complexity issues in this work, the usual approach is to apply an adequate subsampling of the original data [Belabbas and Wolfe, 2009] plus some mechanism to extend the embedding computed on that subsample to the other, not considered data points or, more generally, to new, unseen patterns.

This extension to out-of-sample points is the second challenge in DM and other manifold learning methods. We will consider it here through two different algorithms that extend the diffusion coordinates for new out-of-sample points. The first one is the classical Nyström formula [Bengio et al., 2003] for symmetric, positive semidefinite matrices derived from a kernel that extends the eigenvectors of the sample kernel matrix to the eigenfunctions of the underlying integral operator. We shall see how this method can be easily extended to our transition probability matrix $\mathbf{P}$. The second one, the Laplacian Pyramids (LP) algorithm [Rabin and Coifman, 2012], also relies on a kernel representation but starts from the discrete sample values $f(\mathbf{x}^{(i)})$ (in our case, the eigenvectors of the sample based Markov transition matrix $\mathbf{P}$) of a certain function f (in

our case, the general eigenfunctions), and seeks a multiscale representation of f that allows to approximate the values $f(\mathbf{x})$ from an appropriate multiscale combination of the sample values $f(\mathbf{x}^{(i)})$.

Other classical methods for function extension like Geometric Harmonics [Coifman and Lafon, 2006b] have parameters that need to be carefully set, and in addition there does not exist a robust method of picking the correct neighborhood in the embedding for function smoothing and evaluation. Recently Aizenbud et al. [2014] introduced a geometric PCA-based out-of-sample extension for the purpose of adding new points to a set of already constructed embedding coordinates. A naive way to extend the target function to a new data point could be to find the point's Nearest Neighbors (NN) in the embedded space and average their function values. This NN method for data lifting was compared in Dsilva et al. [2013] with the LP version that was proposed in Rabin and Coifman [2012], and this last method performed better than NN. Buchman et al. [2011] also described a different, point-wise adaptive approach, which requires setting the nearest neighborhood radius parameter for every point.

Nevertheless, and as it is often the case in Machine Learning (ML), when we apply the previous LP model, we can overfit the data if we try to refine too much the predictions during the training phase. In fact, it is difficult to decide when to stop training to obtain good generalization capabilities. A usual approach is to apply the Cross Validation (CV) [Duda et al., 2001, chap. 9] method to get a validation error and to stop when this error starts to increase. An extreme form of CV is the Leave One Out CV (LOOCV): a model is built using all the samples but one, which is then used as a single validation pattern. This is repeated for each sample in the dataset, and the validation error is the average of all the errors. Although LOOCV has a theoretical backing and often yields good results, it has the drawback of a big computational cost.

In this thesis we propose the Auto-adaptative Laplacian Pyramids (ALP), a modification in the LP training algorithm that merges training and approximate LOOCV in one single phase. To do so we simply build the kernel matrix with zeros in its diagonal. As we shall see, with this change we can implement a LOOCV approximation without any additional cost during the training step. This reduces significantly training complexity and provides an automatic criterion to stop training so that we greatly avoid the risk of severe overfitting that may appear in standard LP.

We will explain our adaptive LP proposal in Section 5.3 of this chapter and we will briefly review the standard approaches to Nyström and LP methods in Section 5.2. We will illustrate then their performance over synthetic data (Section 5.4) and over some meteorological datasets (Section 5.5 and Section 5.6).

5.2 Classical Methods

5.2.1 The Nyström Formula

The Nyström formula [Bengio et al., 2003] is a general method to approximate the eigenfunctions $\phi_j(\mathbf{x}^{(i)})$ of a symmetric and positive semidefinite kernel from the eigenvectors $(\phi_j)_i$ of a sample-based kernel matrix, in a way that is consistent with the eigenvalue and eigenvector convergence as the number of sample patterns and, thus, the similarity matrix dimension, grow. In the case of DM, the formula enables to approximately compute the embedding of new patterns without computing again the eigenvalues and eigenvectors of the similarity matrix of the training sample.

The general Nyström method [Bengio et al., 2003] gives us the following expression to extend the eigenvectors of a kernel function:

$$v_j(\mathbf{y}) = \frac{1}{\ell_j} \sum_{i=1}^{n} v_j(\mathbf{x}^{(i)}) \mathcal{K}(\mathbf{y}, \mathbf{x}^{(i)}), \tag{5.1}$$

where $\mathcal{K}(\mathbf{x}, \mathbf{y})$ is a symmetric, positive semi-definite and bounded kernel, and $\{\ell_j\}$, $\{v_j\}$ its eigenvalues and eigenvectors.

This formula comes from the definition of a linear operator $\mathcal{G}f$ associated to our kernel such that [Williams and Seeger, 2001]

$$[\mathcal{G}f](\mathbf{x}) = \int \mathcal{K}(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d\mathbf{y},$$

and that can be expressed in terms of its eigenvalues λ_j and eigenfunctions ϑ_j, verifying

$$[\mathcal{G}\vartheta_j](\mathbf{x}) = \lambda_j \vartheta_j.$$

Discretizing the linear operator and computing a sample-based kernel, it can be proven that the Nyström formula give us approximations to the λ_j and ϑ_j derived from the sample-based eigenvalues and eigenvectors that converge to these general eigenvalues and eigenfunctions of $\mathcal{G}$.

Regarding DM, the problem for applying Equation (5.1) is the symmetric condition over the kernel, as our Markov transition matrix $\mathbf{P}$ does not satisfy this requirement. However, we can apply the Nyström formula to the symmetric kernel a, that was introduced in Section 3.2.2. More precisely if λ_j and $\phi_j(\mathbf{x}^{(i)})$ are the eigenvalues and eigenvectors of the similarity matrix of the sample data, we can extend the eigenvector value for a new point $\mathbf{x}$ as

$$\phi_j(\mathbf{x}) = \frac{1}{\lambda_j} \sum_{i=1}^{N} \phi_j(\mathbf{x}^{(i)}) a(\mathbf{x}, \mathbf{x}^{(i)}).$$

Recalling that

$$\boldsymbol{a}(\mathbf{x}, \mathbf{x}^{(i)}) = \frac{\Pi(\mathbf{x})}{\Pi(\mathbf{y})}\mathbf{P}(\mathbf{x}, \mathbf{x}^{(i)}),$$

and that the eigenvectors ψ_ℓ and φ_ℓ of matrix $\mathbf{P}$ and those of the symmetric kernel $\mathbf{A}$ are related as

$$(\psi_\ell)_i = \frac{(\phi_\ell)_i}{\sqrt{\Pi_i}}$$
$$(\varphi_\ell)_j = (\phi_\ell)_j\sqrt{\Pi_j},$$

the Nyström formula for a new $\mathbf{x}$ yields

$$\begin{aligned}\psi_j(\mathbf{x}) = \frac{\phi_j(\mathbf{x})}{\sqrt{\Pi(\mathbf{x})}} &= \frac{1}{\sqrt{\Pi(\mathbf{x})}}\left(\frac{1}{\lambda_j}\sum_{i=1}^{N}\phi_j(\mathbf{x}^{(i)})\boldsymbol{a}(\mathbf{x}, \mathbf{x}^{(i)})\right)\\ &= \frac{1}{\lambda_j}\sum_{i=1}^{N}\phi_j(\mathbf{x}^{(i)})\frac{\boldsymbol{a}(\mathbf{x}, \mathbf{x}^{(i)})}{\sqrt{\Pi(\mathbf{x})}} = \frac{1}{\lambda_j}\sum_{i=1}^{N}\phi_j(\mathbf{x}^{(i)})\frac{\mathbf{P}(\mathbf{x}, \mathbf{x}^{(i)})}{\sqrt{\Pi\left(\mathbf{x}^{(i)}\right)}}\\ &= \frac{1}{\lambda_j}\sum_{i=1}^{N}\psi_j(\mathbf{x}^{(i)})\mathbf{P}(\mathbf{x}, \mathbf{x}^{(i)}).\end{aligned} \tag{5.2}$$

Thus, the Nyström formula, valid in principle only for symmetric matrices, can be applied almost in its original form to our Markov transition matrix $\mathbf{P}$ and, hence, to extend DM embedding coordinates to new patterns. Moreover, this formulation is also valid for any α parameter value used to compute any of our $\mathbf{P}^{(\alpha)}$ matrices to be used in DM.

The complexity analysis of Nyström's method is easy to make. Observe that to compute Equation (5.2) for a new pattern $\mathbf{x}$ has an $O(N)$ cost for each of the embedding coordinates to be extended. Of course, Equation (5.2) requires the knowledge of the eigenvalues λ_j and eigenvectors $\psi_j(\mathbf{x}^{(i)})$, but these come from the general DM analysis and, thus, do not suppose an extra cost when applying Equation (5.2). In summary, if we use a training sample of size N and work with $\bar{M}$ embedding coordinates, computing these coordinates for a new pattern has a cost $O(\bar{M}\,N)$. We should add to this cost the complexity of computing the distance between the training sample and the new test point, which notice that should be calculated only once for all the embedding coordinate dimensions. This cost will be $O(NM)$, leading to the final Nyström cost which is $O(\bar{M}N) + O(NM) = O(N(\bar{M} + M))$, which is dominated by the $O(NM)$ term, as we will have $M \gg \bar{M}$. We also observe that a nice property of Nyström's method is that it does not need the selection of any algorithm parameter.

5.2.2 The Laplacian Pyramids Algorithm

The Laplacian Pyramids (LP) is an iterative model that was introduced by Burt and Adelson [1983] for its application in image processing and, in particular, image encoding. In its original

version, LP decomposes the input image into a series of images, each of them capturing a different frequency band of the original one. This process is carried out by constructing Gaussian-based smoothing masks of different widths, followed by a down-sampling (quantization) step. LP was later proved to be a tight frame (see Do and Vetterli [2003]) and used for signal processing applications, for example as a reconstruction scheme in Liu et al. [2008].

In Rabin and Coifman [2012], it was introduced a multiscale algorithm in the spirit of LP to be applied in the setting of high-dimensional data analysis. In particular, it was proposed as a simple method for extending low-dimensional embedding coordinates that result from the application of a non-linear dimensionality reduction technique to a high-dimensional data set (a recent application is in Mishne and Cohen [2013]).

We review next the LP procedure as described in Rabin and Coifman [2012] (the down-sampling step, which is part of Burt and Adelson's algorithm, is skipped here). Let $\mathscr{S} = \{\mathbf{x}^{(i)}\}_{i=1}^{N} \in \mathbb{R}^M$ be the sample dataset. The algorithm approximates a function f defined over $\mathscr{S}$ by constructing a series of functions $\{\bar{f}_i\}$ obtained by several refinements d_i over the error approximations. The result of this process gives a function approximation

$$f \simeq \bar{f} = \bar{f}_0 + d_1 + d_2 + d_3 + \cdots$$

In more detail, a first level kernel $\mathcal{K}_\sigma^0(\mathbf{x}, \mathbf{x}') = \Phi\left(\text{dist}(\mathbf{x}, \mathbf{x}')/\sigma\right)$ is chosen using a wide, initial scale σ and where $\text{dist}(\mathbf{x}, \mathbf{x}')$ denotes some distance function between points in the original dataset. A usual choice and the one we will use here is the Gaussian kernel with Euclidean distances, i.e., to take $\text{dist}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|$ and then define

$$\mathcal{K}^0(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \kappa e^{-\frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2}{\sigma^2}},$$

where κ is the Gaussian kernel normalizing constant.

The smoothing operator $\mathcal{G}^0$ is constructed as a normalization of the initial kernel, in the following form

$$\mathcal{G}^0(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \frac{\mathcal{K}^0(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})}{\sum_k \mathcal{K}^0(\mathbf{x}^{(i)}, \mathbf{x}^{(k)})}. \tag{5.3}$$

A first coarse representation of f is then generated by the convolution $\bar{f}_0 = f * \mathcal{G}^0$ that captures the low-frequencies of the function.

For the next iteration levels we fix a $\mu > 1$, and construct at level i a sharper Gaussian kernel $\mathcal{G}^i$ with scale σ/μ^i in two steps as done before: we compute first the Gaussian kernel $\mathcal{K}^i$ using σ/μ^i and we normalize it to obtain the new smoothing operator $\mathcal{G}^i$. In each level, the residual

$$d_{i-1} = f - \bar{f}_{i-1},$$

which captures the error of the approximation to f at the previous $i-1$ step, is used to generate a more detailed representation of f given by

$$\bar{f}_i = \bar{f}_{i-1} + d_{i-1} * \mathcal{G}^i = \bar{f}_{i-1} + h_{i-1},$$

with $h_\ell = d_\ell * \mathcal{G}^{\ell+1}$. The iterative algorithm stops once the norm of d_i residual vector is smaller than a predefined error. Stopping at iteration L, the final LP model has thus the form

$$\bar{f}_L = \bar{f}_0 + \sum_0^{L-1} h_\ell = f * \mathcal{G}^0 + \sum_0^{L-1} d_\ell * \mathcal{G}^{\ell+1},$$

and extending this multiscale representation to a new data point $\mathbf{x} \in \mathbb{R}^M$ is now straightforward because we simply set

$$\begin{aligned}\bar{f}_L(\mathbf{x}) &= f * \mathcal{G}^0(\mathbf{x}) + \sum_0^{L-1} d_\ell * \mathcal{G}^{\ell+1}(\mathbf{x}) \\ &= \sum_j f_j \mathcal{G}^0(\mathbf{x}, \mathbf{x}^{(j)}) + \sum_1^{L} \sum_j d_{\ell-1;j} \mathcal{G}^\ell(\mathbf{x}, \mathbf{x}^{(j)}),\end{aligned}$$

where we directly define the $\mathcal{G}^\ell$ kernel for a new $\mathbf{x}$ as

$$\mathcal{G}^\ell(\mathbf{x}, \mathbf{x}^{(j)}) = \frac{\mathcal{K}^\ell(\mathbf{x}, \mathbf{x}^{(j)})}{\sum_k \mathcal{K}^\ell(\mathbf{x}, \mathbf{x}^{(k)})}.$$

5.2.2.1 Complexity Analysis for the LP Algorithm

The overall cost of the LP algorithm is easy to analyze. During training, computing the convolutions $\bar{f}_0 = f * \mathcal{G}^0$ and $h_\ell = d_\ell * \mathcal{G}^{\ell-1}$ has a $O(\bar{M}N^2)$ cost for a N-size sample per each new dimension, while that of obtaining the d_ℓ is just $O(N)$ per step in LP. Thus, the cost of L LP steps is $O(L\bar{M}N^2)$. To this cost we should add the distance computation, which for the training sample will be of order $O(MN^2)$.

With respect to the cost of LP in testing, it is $O(L\bar{M}N)$ for extending the coordinates plus $O(MN)$ for computing the distances between a test pattern and those in the sample. The overall cost of the LP algorithm for one test point depends thus on the relationship between $L\bar{M}$ and M. In our experiments, we will have $M \gg L\bar{M}$, and the main cost in training will be dominated by $O(MN^2)$ and in test by $O(MN)$.

We observe that if we set a very small error threshold and run afterwards enough iterations, we will end up having $\bar{f}_\ell = f$ over the training sample. In fact, $\bar{f}_\ell = \bar{f}_{\ell-1} + h_{\ell-1}$ and, therefore,

$$\begin{aligned}\bar{f}_\ell &= \bar{f}_{\ell-1} + h_{\ell-1} = \bar{f}_{\ell-1} + (f - \bar{f}_{\ell-1}) * \mathcal{G}^\ell \\ &= f * \mathcal{G}^\ell + \bar{f}_{\ell-1} * (\mathbf{I} - \mathcal{G}^\ell),\end{aligned}$$

with $\mathbf{I}$ denoting the identity matrix. Now, if we have $\bar{f}^\ell \to \phi$, it follows taking limits that

$$\phi = f * \lim \mathcal{G}^\ell + \phi * \lim(\mathbf{I} - \mathcal{G}^\ell)$$

i.e., $\phi = f$, for $\mathcal{G}^\ell \to \mathbf{I}$.

In practice, we will numerically have $\mathcal{G}^\ell = \mathbf{I}$ as soon as ℓ is large enough so that we have $\mathcal{K}^\ell(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \simeq 0$, and thus $\mathcal{G}^\ell(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \simeq 0$. We then have $d_{\ell;j} = 0$ for all j and the LP model does not change anymore. In other words, care has to be taken when deciding to stop the LP iterations to avoid overfitting. In fact, we show next that when using Gaussian kernels, as we do, the ℓ_2 norm of the LP errors $\hat{d}_\ell$ decay extremely fast.

5.2.2.2 Error Analysis for the LP Scheme

For analyzing the LP error, first notice that working in the continuous kernel setting, we have $\mathcal{G} = \mathcal{K}$ for a Gaussian kernel where the denominator in Equation (5.3) is just $\int \mathcal{K}(\mathbf{x}, \mathbf{z})d\mathbf{z} = 1$. Assume that f is in L_2, so $\int_{\mathbf{x}} f^2(\mathbf{x})\, d\mathbf{x} < \infty$. The LP scheme is a relaxation process for which in the first step the function f is approximated by $\mathcal{H}^0(f) = f * \mathcal{G}^0(\mathbf{x})$. For all ℓ, $\mathcal{G}^\ell(\mathbf{x})$ is an approximation to a delta function satisfying

$$\begin{aligned} \int \mathcal{G}^\ell(\mathbf{x})\, d\mathbf{x} &= 1, \\ \int \mathbf{x}\mathcal{G}^\ell(\mathbf{x})\, d\mathbf{x} &= 0, \\ \int \mathbf{x}^2\mathcal{G}^\ell(\mathbf{x})\, d\mathbf{x} &\leq 2c. \end{aligned} \tag{5.4}$$

In the second step f is approximated by $\mathcal{H}^0(f) + \mathcal{H}^1(d_0)$, where $d_0 = \mathcal{H}^0(f) - f$ and $\mathcal{H}^1(d_0) = d_0 * \mathcal{G}^1(\mathbf{x})$, and so on. Taking the Fourier transform of $\mathcal{G}^\ell(\mathbf{x})$, we have (see Fishelov [1990])

$$\left|\hat{\mathcal{G}}^\ell(\omega) - 1\right| \leq \frac{\sigma^2}{2}\int \mathbf{x}^2\mathcal{G}^\ell(\mathbf{x})\, dx \leq c\sigma^2, \tag{5.5}$$

where we have used Equation (5.4).

We first analyze the error $d_0(\mathbf{x})$ in the first step, which is defined by $d_0(\mathbf{x}) = f * \mathcal{G}^0(\mathbf{x}) - f$. Taking the Fourier transform of $d_0(\mathbf{x})$ and using Equation (5.5) we have

$$\left|\hat{d}_0(\omega)\right| = \left|\hat{f}(w)\right|\left|\hat{\mathcal{G}}^0(\omega) - 1\right| \leq c\sigma_0^2\left|\hat{f}(\omega)\right|. \tag{5.6}$$

The error in the second step is

$$d_1(x) = d_0 - \mathcal{H}_1(d_0) = \left(f * \mathcal{G}^0 - f\right) - d_0 * \mathcal{G}^1. \tag{5.7}$$

Taking the Fourier transform of Equation (5.7) yields

$$\begin{aligned}\left|\hat{d}_1(\omega)\right| &= \left|\hat{d}_0(\omega) - \hat{d}_0(\omega)\hat{\mathcal{G}}^1(\omega)\right| \\ &= \left|\hat{d}_0(\omega)\right|\left|\hat{\mathcal{G}}^1(\omega) - 1\right|.\end{aligned}$$

Using Equation (5.5) and Equation (5.6) we obtain

$$\left|\hat{d}_1(\omega)\right| \leq c\left|\hat{d}_0(\omega)\right|\sigma_1^2 \leq c\sigma_0^2\sigma_1^2\left|\hat{f}(\omega)\right|.$$

Since $\sigma_1 = \frac{\sigma_0}{\mu}$ with $\mu > 1$, then $\left|\hat{d}_1(\omega)\right| \leq c\sigma_0^2\frac{\sigma_0^2}{\mu^2}\left|\hat{f}(\omega)\right|$. Similarly, for the ℓ-th step the error is bounded by

$$\left|\hat{d}_\ell(\omega)\right| \leq c\sigma_0^2\left(\frac{\sigma_0}{\mu^\ell}\right)^2\left|\hat{f}(\omega)\right| = c\frac{\sigma_0^4}{\mu^{2\ell}}\left|\hat{f}(\omega)\right|.$$

By Parseval's equality we obtain

$$\left\|d_\ell\right\|_{L^2} \leq c\frac{\sigma_0^4}{\mu^{2\ell}}\left\|f\right\|_{L^2}.$$

Thus, the ℓ_2 error has a very fast linear decay rate.

5.3 The Auto-adaptative Laplacian Pyramids Algorithm

The standard way to prevent overfitting is to use an independent validation subset and to stop the above ℓ iterations as soon as the validation error on that subset starts to increase. This can be problematic for small samples and introduces a random dependence on the choice of the particular validation subset. k-fold CV is then usually the standard choice, in which we randomly distribute the sample in k subsets, and iteratively use $k-1$ subsets for training and the remaining one for validation. In the extreme case when $k = N$, i.e., we use just one pattern for validation, we arrive at LOOCV and stop the training iterations when the LOOCV error starts to increase. Besides its simplicity, LOOCV has the attractive of being an almost unbiased estimator of the true generalization error (see for instance [Cawley and Talbot, 2004; Elisseeff and Pontil, 2002]), although with possibly a high variance [Kohavi, 1995]. In our case LOOCV can be easily applied using for training a $N \times N$ normalized kernel matrix $\mathcal{G}_{(p)}$ which is just the previous matrix $\mathcal{G}$ where we set to 0 the p-th rows and columns when $\mathbf{x}^{(p)}$ is held out of the training sample and used for validation. The most obvious drawback of LOOCV is its rather high cost, which in our case is $N \times O(LN^2) = O(LN^3)$ cost. However, it is often the case for other models that there are ways to estimate the LOOCV error with a smaller cost. This can be done exactly in the case of κ-Nearest Neighbors (κ-NN) [Fukunaga and Hummels, 1989] or of Ordinary Linear Least Squares (OLS) [Hastie et al., 2008, Chapter 7], or approximately for

Support Vector Machines (SVM) [Chapelle et al., 2002] or Gaussian Processes [Rasmussen and Williams, 2005].

In order to alleviate this cost here, notice first that when we removed $\mathbf{x}^{(p)}$ from the training sample, the test value at $\mathbf{x}^{(p)}$ of the $f^{(p)}$ extension built is

$$\begin{aligned} f_L^{(p)}(\mathbf{x}^{(p)}) &= \sum_{j\neq p} f_j \mathcal{G}^0(\mathbf{x}^{(p)}, \mathbf{x}^{(j)}) + \sum_{\ell=1}^{L} \sum_{j\neq p} d_{\ell-1;j}^{(p)} \mathcal{G}^\ell(\mathbf{x}^{(p)}, \mathbf{x}^{(j)}) \\ &= \sum_{j} f_j \tilde{\mathcal{G}}^0(\mathbf{x}^{(p)}, \mathbf{x}^{(j)}) + \sum_{\ell=1}^{L} \sum_{j} d_{\ell-1;j}^{(p)} \tilde{\mathcal{G}}^\ell(\mathbf{x}^{(p)}, \mathbf{x}^{(j)}), \end{aligned}$$

where $d_\ell^{(p)}$ are the previously defined different residuals computed using the $\mathcal{G}_{(p)}^\ell$ matrices and where $\tilde{\mathcal{G}}$ is now just the normalized kernel $\mathcal{G}$ with its diagonal elements set to 0, i.e. $\tilde{\mathcal{G}}_{i,i} = 0$, $\tilde{\mathcal{G}}_{i,j} = \mathcal{G}_{i,j}$ when $j \neq i$.

This observation leads to the modification we propose on the standard LP algorithm given in Rabin and Coifman [2012], and which simply consist in applying the LP procedure previously described in Section 5.2.2 but replacing the $\mathcal{G}$ matrix by its 0-diagonal version $\tilde{\mathcal{G}}$, computing then $\tilde{f}_0 = f * \tilde{\mathcal{G}}^0$ at the beginning, and $\tilde{d}_\ell = f - \tilde{f}_\ell$, $\tilde{h}_\ell = \tilde{d}_\ell * \tilde{\mathcal{G}}^{\ell+1}$ and $\tilde{f}_\ell$ vectors at each iteration. We call this algorithm the Auto-adaptative Laplacian Pyramids (ALP).

According to the previous formula for the $f_L^{(p)}(\mathbf{x}^{(p)})$, we can take the ALP values $\tilde{f}_{L,p} = \tilde{f}_L(\mathbf{x}^{(p)})$ given by

$$\tilde{f}_L(\mathbf{x}^{(p)}) = \sum_{j} f_j \tilde{\mathcal{G}}^0(\mathbf{x}^{(p)}, \mathbf{x}^{(j)}) + \sum_{\ell=1}^{L} \sum_{j} \tilde{d}_{\ell-1;j} \tilde{\mathcal{G}}^\ell(\mathbf{x}^{(p)}, \mathbf{x}^{(j)}),$$

as approximations to the LOOCV validation values $f_L^{(p)}(\mathbf{x}^{(p)})$. But then we can approximate the square LOOCV error at iteration L as

$$\sum_p (f(\mathbf{x}^{(p)}) - f_L^{(p)}(\mathbf{x}^{(p)}))^2 \simeq \sum_p (f(\mathbf{x}^{(p)}) - \tilde{f}_{L,p})^2 = \sum_p (\tilde{d}_{L;p})^2,$$

which is just the training error of ALP at the current iteration. In other words, working with the $\tilde{\mathcal{G}}$ matrix instead of $\mathcal{G}$, the training error at step L gives in fact an approximation to the LOOCV error at this step. The cost of running L steps of ALP is now just $O(LN^2)$ and, thus, we gain the advantage of the exhaustive LOOCV without any additional cost on the overall algorithm. The complete training procedure is presented in Algorithm 5.1 and the test algorithm is shown in Algorithm 5.2.

The obvious advantage of ALP is that when we evaluate the training error, we are actually estimating approximately the LOOCV error after each LP iteration. Therefore, the evolution of these LOOCV values tells us which is the optimal iteration to stop the algorithm, i.e., just when the training error approximation to the LOOCV error starts growing. Thus, we do not

ALGORITHM 5.1: Auto-adaptative Laplacian Pyramids Training Algorithm.

Input: $\mathscr{D}_{\text{tr}}$, $\mathbf{y}_{\text{tr}}$, σ_0, μ ;

Output: $(\{d_i\}, \sigma_0, \mu, k)$, the training model ;

Initialization: $\sigma = \sigma_0$; $d_0 = y_{\text{tr}}$;

Initialization: $\tilde{f}_0 = 0$; $i = 1$;

while $(\text{err}_i < \text{err}_{i-1})$ **do**

  $\mathcal{K} = e^{-\|\mathscr{D}_{\text{tr}} - \mathscr{D}_{\text{te}}\|^2/\sigma^2}$;

  $\mathcal{G}_i \leftarrow \text{normalize}(\mathcal{K})$;

  $\tilde{\mathcal{G}} = \mathcal{G}$ with 0-diagonal ; ▶ *LOOCV.*

  $\tilde{f}_i = \tilde{f}_{i-1} + d_{i-1} * \tilde{\mathcal{G}}_i$;

  $d_i = f - \tilde{f}_i$;

  $\text{err}_i = d_i/_{P_{\text{tr}}}$, with P_{tr} the number of patterns in $\mathscr{D}_{\text{tr}}$;

  $\sigma = \sigma/_{\mu}$; $i = i + 1$;

end while

$k = \min_i\{d_i\}.$; ▶ *Optimal iteration.*

ALGORITHM 5.2: Auto-adaptative Laplacian Pyramids Testing Algorithm.

Input: $\mathscr{D}_{\text{tr}}$, $\mathscr{D}_{\text{te}}$, σ_0, $(\{d_i\}, \sigma_0, \mu, k)$;

Output: $\hat{\mathbf{y}}_{\text{te}}$;

Initialization: $\hat{\mathbf{y}}_{\text{te}} = 0$; $\sigma = \sigma_0$;

for $i = 0, \ldots, k-1$ **do**

  $\mathcal{K} = e^{-\|\mathscr{D}_{\text{tr}} - \mathscr{D}_{\text{te}}\|^2/\sigma^2}$;

  $\mathcal{G}_i \leftarrow \text{normalize}(\mathcal{K})$;

  $\hat{\mathbf{y}}_{\text{te}} = \hat{\mathbf{y}}_{\text{te}} + d_i * \mathcal{G}_i$;

  $\sigma = \sigma/_{\mu}$;

end for

only remove the danger of overfitting but can also use the training error as an approximation to the generalization error.

Moreover, ALP achieves an automatic selection of the width of the Gaussian kernel which makes this version of LP to be autoadaptative as it does not require costly parameter selection procedures. In fact, choosing as customarily done $\mu = 2$, the only required parameter would be the initial σ but provided it is wide enough, its $\sigma/2^{\ell}$ scaling will yield an adequate final kernel width.

5.3.1 Possible Applications of ALP

As explained in Section 5.1, one of the biggest challenges when working with DM is to apply them to new, unseen patterns without having to recompute the eigenvalue and eigenvector structure over which DM rely. The Nyström formula and the LP algorithm appear as suitable tools for this goal and we will study here how they perform and how they compare in some examples. We have also presented in this chapter an autoadaptative modification on the LP model (Section 5.3) which is not only useful for diffusion coordinates extension but also for general regression problems.

More concretely, for a better understanding of this theory and its advantages, we will first illustrate in Section 5.4 the proposed ALP algorithm over a synthetic example of a composition of sines with different frequencies plus additive noise.

We have studied the DM-coordinate extension in real data problems, as it the case of the meteorological wind data presented in Section 3.3 and of some solar radiation forecasted data provided by the AMS Kaggle Competition. For both examples we will show in Section 5.5 and Section 5.6 how the different methods compare for extending diffusion coordinates.

5.4 ALP Behavior on a Synthetic Example

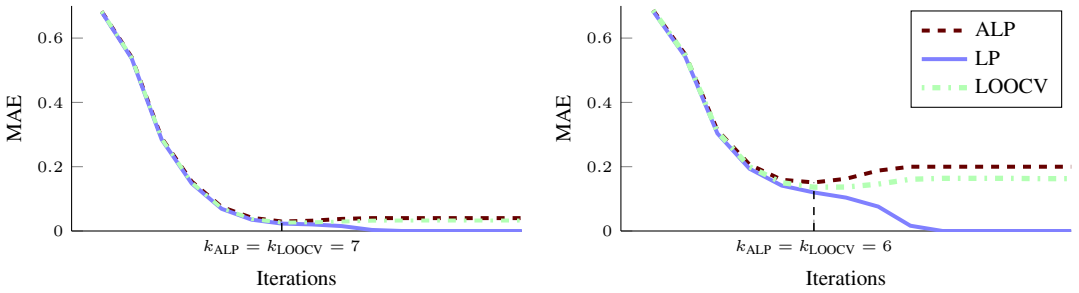
For a better understanding of the theory exposed in this chapter, we first illustrate the proposed ALP algorithm on a synthetic example of a composition of sines with different frequencies plus additive noise.

We consider a sample $\mathbf{x}$ with N points equally spaced over the range $[0, 10\pi]$. The target function f is then

$$f = \sin(\mathbf{x}) + 0.5\sin(3\mathbf{x}) \cdot \iota_2(\mathbf{x}) + 0.25\sin(9\mathbf{x}) \cdot \iota_3(\mathbf{x}) + \varepsilon,$$

where ι_2 is the indicator function of the interval $(10\pi/3, 10\pi]$, ι_3 that of $(2 \cdot 10\pi/3, 10\pi]$ and $\varepsilon \sim \mathcal{U}([-\delta, \delta])$ is uniformly distributed noise. In other words, we have a single frequency in the interval $[0, 10\pi/3]$, two frequencies in $(10\pi/3, 2 \cdot 10\pi/3]$ and three in $(2 \cdot 10\pi/3, 10\pi]$. We run two different simulations, the first one with $4,000$ points with small $\delta = 0.05$ noise and the second one with $2,000$ points and a larger $\delta = 0.25$ (observe that $|f| \leq 1.75$). In both cases, odd indexed points form the training set and even points form the test set.

Recall that we have emphasized that the main advantage of ALP is that when we evaluate the training error, we also obtain an approximate estimate of the LOOCV error after each LP iteration, and we should stop when the error training starts growing, removing the danger of overfitting.



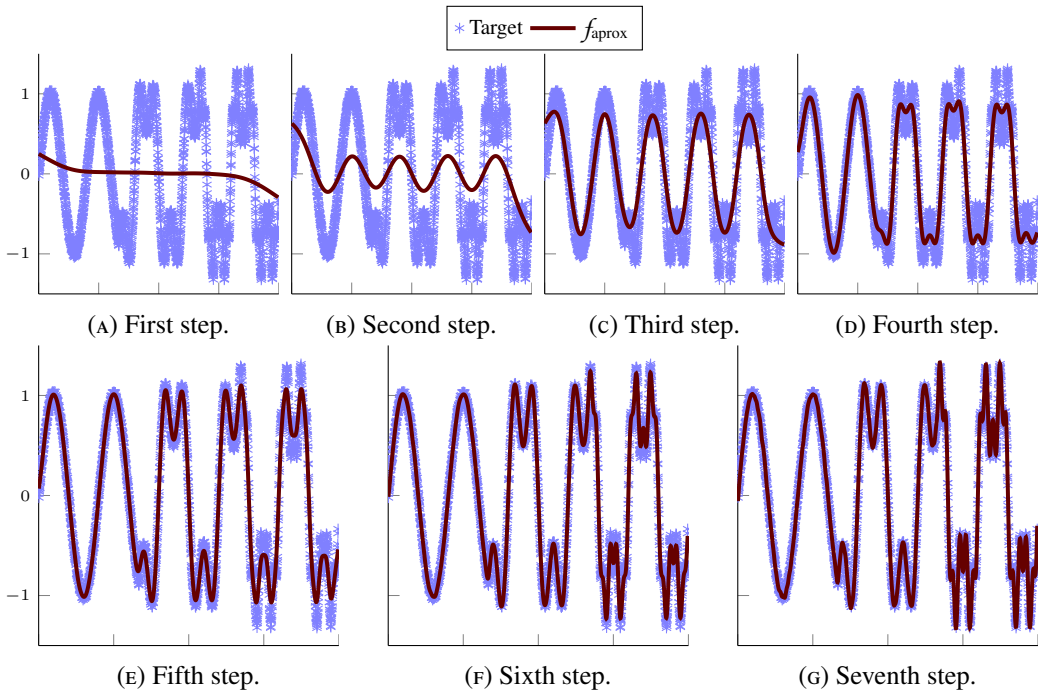
(A) Less Noisy Example. (B) Noisier Example.

FIGURE 5.1: Training and LOOCV errors for the original and modified LP models applied over a perturbed sine.

This effect can be observed in Figure 5.1 when our LP proposal is applied to this synthetic example. The solid blue and dashed black lines represent the LP training error and the LOOCV error per iteration respectively, and the dashed blue line represents the error for our proposed method. The blue line, that corresponds to the ALP training error attains its minimum at the same iteration prescribed by LOOCV for LP.

The ALP model is also able to automatically adapt its multiscale behavior to the data, trying to refine the prediction in each iteration using a more localized kernel, given by a smaller σ. This behavior can be observed in Figure 5.2, which shows the evolution of the ALP prediction for the small noise experiment. As we can see, at the beginning, the model approximates the function just by a coarse mean of the target function values; however, in the subsequent iterations that start using sharper kernels and refined residuals, the approximating function starts capturing the different frequencies and amplitudes of the composite sines. In this particular case the minimum LOOCV value is reached after 7 iterations (see Figure 5.1), a relatively small number which makes sense as we have a simple model with small noise.

When we repeat the same experiment but now enlarging the amplitude of the uniform distribution to $\delta = 0.25$, the predicted function is represented in Figure 5.3 and it is obtained after 6 iterations (as shown in the right picture of Figure 5.1). As it was to be expected, the number of LP iterations is now slightly smaller than in the previous example because the algorithm selects a more conservative, smoother prediction in the presence of noisier and, thus, more difficult data. In any case, we can conclude that the ALP model captures very well the essential behavior underlying both samples, catching the three different frequencies of the sine and their amplitudes even when the noise level increases.



(A) First step. (B) Second step. (C) Third step. (D) Fourth step.

(E) Fifth step. (F) Sixth step. (G) Seventh step.

FIGURE 5.2: Evolution of the Auto-adaptative Laplacian Pyramids model for the first example.

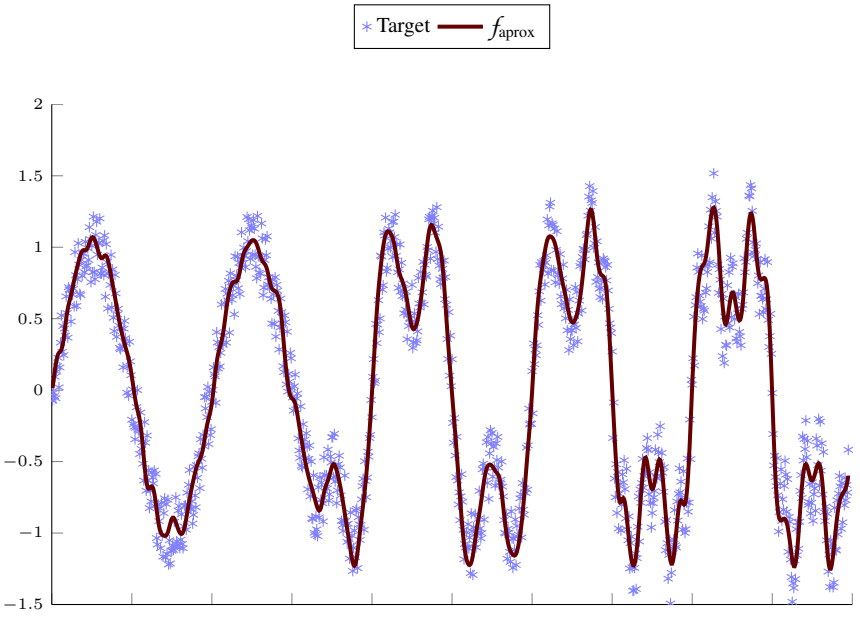


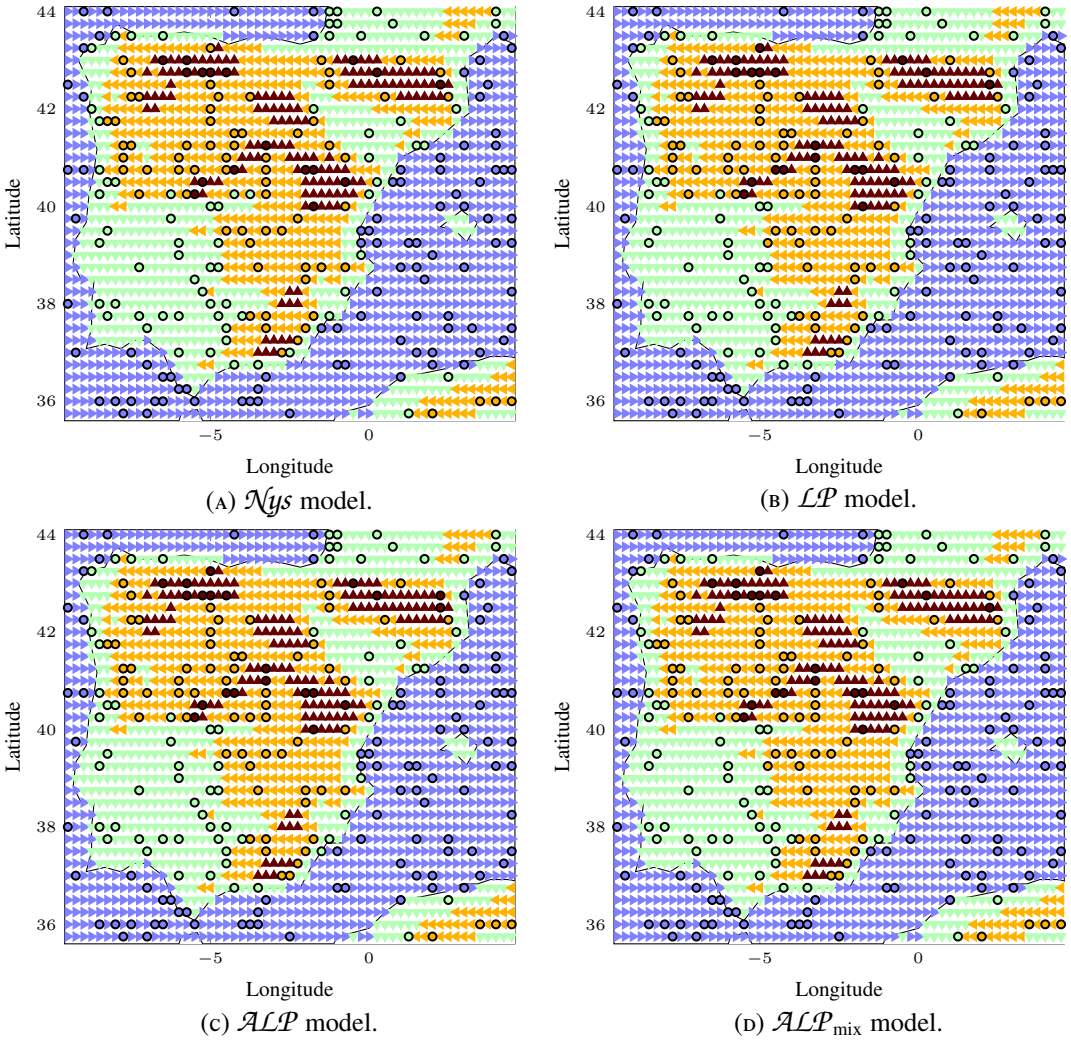
FIGURE 5.3: Prediction of the Auto-adaptative Laplacian Pyramids model for a more noisy sine.

5.5 ALP in NWP Time Compression

We would like to present next a comparison between the Nyström formula, the classical LP method and the new proposed ALP for extending diffusion coordinates. We want to apply these algorithms first on the example presented in Section 3.3 for time compression of weather forecasting data, where the results obtained can be easily visualized over the Spanish map. Recall that, for this problem we have considered for each one of the $1,995$ nodes in the ECMWF grid for the Iberian Peninsula a feature vector made up by the 5 variable surface forecasts at that node for the whole year and the 8 three-hour snapshots of meteorological data that ECMWF generates per day. We recall that we assigned to each node a vector with dimension $5\times8\times365 = 14,600$, and saw in that chapter that we can compress that feature vector into another one of a much lower dimension in a way that still captures meaningful information for each node. For our comparison in this section we will randomly take out of the sample three subsets of one-year patterns with 100, 250 and 500 grid nodes respectively (i.e., approximately between 5% and 25% of the original sample) that, in turn, will form the test datasets.

We point out that, due to the randomness involved in the selection of the test subset, we will do 100 different experiments with each model. In all the cases, we have first computed the DM eigenstructure over the new training set with parameter values $\alpha = 1$ and $t = 1$ for then extend them to test patterns, obtaining the new embedding coordinates using an out-of-sample method. We present in this example the comparison between four different out-of-sample models: the Nyström method that we will call $\mathcal{Nys}$, the classical LP method named $\mathcal{LP}$, the proposed ALP method denoted as $\mathcal{ALP}$, and a mixture between LP and ALP which consist in using at the same time the ALP approach (i.e., with zeros in the diagonal of the kernel matrix) to determine when to stop the algorithm and the LP approach (i.e., with a non-zero diagonal) to build the final model. This alternative model is called $\mathcal{ALP}_{\text{mix}}$, and it is presented for comparison purposes, as it is supposed to be thew best model if $\mathcal{LP}$ outperforms the proposed $\mathcal{ALP}$ model, but will still be faster than the classical method. Recall also that the classical LP training algorithm has a stopping criterion based on the number of iterations that we should prefixed. To obtain an optimal value for this parameter we have computed a 5-fold cross-validation.

To evaluate and compare the different methods, we will get a first visual measure of the quality of the out-of-sample extension applying κ-means with $\kappa = 4$ to the embedded coordinates of the training points, and then associating the test extended embedding coordinates to the closest training cluster. If the out-of-sample embeddings are correct, the test patterns should be assigned to the same clusters of their neighbors. We can visually appreciate whether this is the case checking that the cluster color of the test points coincides with the cluster color of their training neighbors. Figure 5.4 shows the results obtained with each method for one of the experiments done with a random subset of 250 test points which have been marked by circles to differentiate them from the training points depicted as triangles. As we can see, the cluster coincidence



(A) $\mathcal{N}ys$ model. (B) $\mathcal{LP}$ model.

(C) $\mathcal{ALP}$ model. (D) $\mathcal{ALP}_{\text{mix}}$ model.

FIGURE 5.4: Extension over the Iberian Peninsula of the diffusion coordinates for new weather patterns in the time compression example discussed in Section 3.3. Test grid points appear as circles and are colored by cluster assignments computed over their extended embedding coordinates.

between test points and their neighbors happens most of the time and when this is not the case, the errors can be considered as relatively minor as the cluster assignment is still coherent with a closer node, and for example, a sea point is never taken as a mountain one.

A better, more precise way of measuring the accuracy of Nyström and LP projections is to compare the cluster assignments of the test points after applying κ-means on the training subgrid (the "predicted" assignments) with those made when projections are computed after applying DM to the entire grid without removing any node (the "real" assignments). In other words, we transform the out-of-sample computations into a classification problem, where a test point is correctly classified if the cluster to which it is assigned according to its embedded coordinates coincides with the one it belongs when κ-means is applied over the DM coordinates computed over the entire grid. The classification accuracy, that is, the percentage of test nodes assigned

TABLE 5.1: Confusion matrices of the out-of-sample extensions for the average over the 100 experiments with 100-size test subsets.

	P_1	P_2	P_3	P_4	$\sum$
R_1	39.93	0.49	0.00	0.00	40.42
R_2	0.56	25.07	0.97	0.00	26.60
R_3	0.00	1.07	24.25	0.50	25.82
R_4	0.00	0.00	0.54	6.62	7.16
$\sum$	40.49	27.17	25.76	7.12	100.00

(A) $\mathcal{N}ys$ model. Accuracy: 95.87%

	P_1	P_2	P_3	P_4	$\sum$
R_1	39.98	0.44	0.00	0.00	40.42
R_2	0.15	26.17	0.28	0.00	26.60
R_3	0.00	0.03	25.79	0.00	25.82
R_4	0.00	0.00	0.56	6.60	7.16
$\sum$	40.13	27.20	26.63	6.60	100.00

(B) $\mathcal{LP}$ model. Accuracy: 98.54%

	P_1	P_2	P_3	P_4	$\sum$
R_1	40.32	0.10	0.00	0.00	40.42
R_2	0.07	26.45	0.08	0.00	26.60
R_3	0.00	0.01	25.72	0.09	25.82
R_4	0.00	0.00	0.00	7.16	7.16
$\sum$	40.39	26.56	25.80	7.25	100.00

(C) $\mathcal{ALP}$ model. Accuracy: 99.65%

	P_1	P_2	P_3	P_4	$\sum$
R_1	40.26	0.16	0.00	0.00	40.42
R_2	0.15	26.29	0.16	0.00	26.60
R_3	0.00	0.02	25.80	0.00	25.82
R_4	0.00	0.00	0.37	6.79	7.16
$\sum$	40.41	26.84	26.33	6.79	100.00

(D) $\mathcal{ALP}_{\text{mix}}$ model. Accuracy: 99.14%

to their "real" clusters, can be used as quality measure. Table 5.1, Table 5.2 and Table 5.3 include the confusion matrices of the fourth methods for the different test set sizes. These matrices confirm the overall good performance, and also that the few missclassifications are always between near clusters. Notice that the clusters will be numbered accordingly to their mean geopotential altitude, thus, cluster 1 will represent the sea and cluster 4, the mountains. Notice also that the results in these tables correspond to the averages computed over the 100 experiments repeated with different random test subsets. It can be appreciated that all methods offer a very similar high accuracy for each dataset. It should be remarked that the proposed $\mathcal{ALP}$ and $\mathcal{ALP}_{\text{mix}}$ methods outperform the others in the case of the experiments with 100 and 250 test points. For the most complicated example, the subset with 500 test points—around the 25% of the original sample— the Nyström method is slightly better than the other ones.

Notice, however, that we have just proved that we obtain a good clustering accuracy when we extend the diffusion coordinates with these methods. This is thus an indirect way of measuring the quality of the projections. We would also like to measure in a more intrinsic way that these algorithms give us a good approximate embedding coordinates for new points. A simple way of doing so is to compute what we may call the Frobenius distance, that is, the Frobenius norm of the data matrix with the differences between the exact, i.e., the "true" embedding coordinates of the test points and their test coordinates computed using an out-of-sample method. We recall that the Frobenius norm of a matrix is just the Euclidean norm of the vectorized matrix entries, i.e. $\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$. We use a relative Frobenius distance defined as $\text{dis}_F = \|\mathbf{DM}_{\text{full}} - \mathbf{DM}_{\text{extd}}\|_F / \|\mathbf{DM}_{\text{full}}\|_F$, where $\mathbf{DM}_{\text{full}}$ represents the DM coordinates of the test points obtained applying DM over the training and test sets together, and $\mathbf{DM}_{\text{extd}}$ represents the coordinates where DM has been computed over the training set and those points relative to

TABLE 5.2: Confusion matrices of the out-of-sample extensions for the average over the 100 experiments with 250-size test subsets.

	P_1	P_2	P_3	P_4	$\sum$
R_1	98.55	0.78	0.00	0.00	99.33
R_2	1.48	65.94	1.79	0.00	69.21
R_3	0.00	1.65	60.83	0.83	63.31
R_4	0.00	0.00	0.90	17.25	18.15
$\sum$	100.03	69.27	63.52	18.08	250.00

(A) $\mathcal{Nys}$ model. Accuracy: 97.03%

	P_1	P_2	P_3	P_4	$\sum$
R_1	99.16	0.77	0.00	0.00	99.93
R_2	0.74	67.97	0.46	0.00	69.17
R_3	0.00	0.07	63.00	0.01	63.08
R_4	0.00	0.00	1.33	16.49	17.82
$\sum$	99.90	70.14	64.79	16.50	250.00

(B) $\mathcal{LP}$ model. Accuracy: 98.65%

	P_1	P_2	P_3	P_4	$\sum$
R_1	99.14	0.19	0.00	0.00	99.33
R_2	0.90	68.00	0.26	0.00	69.16
R_3	0.00	0.74	62.18	0.43	63.35
R_4	0.00	0.00	0.47	17.69	18.16
$\sum$	100.04	69.40	62.91	18.12	250.00

(C) $\mathcal{ALP}$ model. Accuracy: 98.80%

	P_1	P_2	P_3	P_4	$\sum$
R_1	99.59	0.34	0.00	0.00	99.93
R_2	0.55	68.33	0.29	0.00	69.17
R_3	0.00	0.05	63.01	0.02	63.08
R_4	0.00	0.00	1.00	16.82	17.82
$\sum$	100.14	69.72	64.30	16.84	250.00

(D) $\mathcal{ALP}_{\text{mix}}$ model. Accuracy: 99.10%

TABLE 5.3: Confusion matrices of the out-of-sample extensions for the average over the 100 experiments with 500-size test subsets.

	P_1	P_2	P_3	P_4	$\sum$
R_1	198.34	1.25	0.00	0.00	199.59
R_2	1.89	132.56	2.59	0.00	137.04
R_3	0.00	2.36	122.53	1.53	126.42
R_4	0.00	0.00	2.68	34.27	36.95
$\sum$	200.23	138.85	127.80	35.80	500.00

(A) $\mathcal{Nys}$ model. Accuracy: 97.54%

	P_1	P_2	P_3	P_4	$\sum$
R_1	193.07	6.52	0.00	0.00	199.59
R_2	1.49	128.46	7.09	0.00	137.04
R_3	0.00	0.69	122.40	3.33	126.42
R_4	0.00	0.00	4.02	32.93	36.95
$\sum$	194.56	139.69	133.51	36.26	500.00

(B) $\mathcal{LP}$ model. Accuracy: 95.37%

	P_1	P_2	P_3	P_4	$\sum$
R_1	196.95	2.64	0.00	0.00	199.59
R_2	0.96	131.79	4.29	0.00	137.04
R_3	0.00	0.76	122.73	2.93	126.42
R_4	0.00	0.00	1.46	35.49	36.95
$\sum$	197.91	136.65	128.48	38.42	500.00

(C) $\mathcal{ALP}$ model. Accuracy: 97.39%

	P_1	P_2	P_3	P_4	$\sum$
R_1	193.80	5.79	0.00	0.00	199.59
R_2	1.11	130.30	5.63	0.00	137.04
R_3	0.00	0.63	123.26	2.53	126.42
R_4	0.00	0.00	3.07	33.88	36.95
$\sum$	194.91	139.79	131.96	36.41	500.00

(D) $\mathcal{ALP}_{\text{mix}}$ model. Accuracy: 96.25%

the test subset have been approximated via out-of-sample models. Of course, this Frobenius distance will depend on the number of test points considered.

The Frobenius distance can be measured over the training points or over the test points, and considering both distances make sense in this context. Recall that embeddings are computed after a suitable eigenanalysis of the training subsample, which is quite influenced by the selection made of the test subset, particularly when, as it will be the case in some examples, it is a sizable part of the entire sample. Notice that if the embeddings of the training subsample are quite different when computed over the full matrix or over the training submatrix, they will also

TABLE 5.4: Median relative Frobenius distances between exact and approximated embedding coordinates computed for 100 experiments.

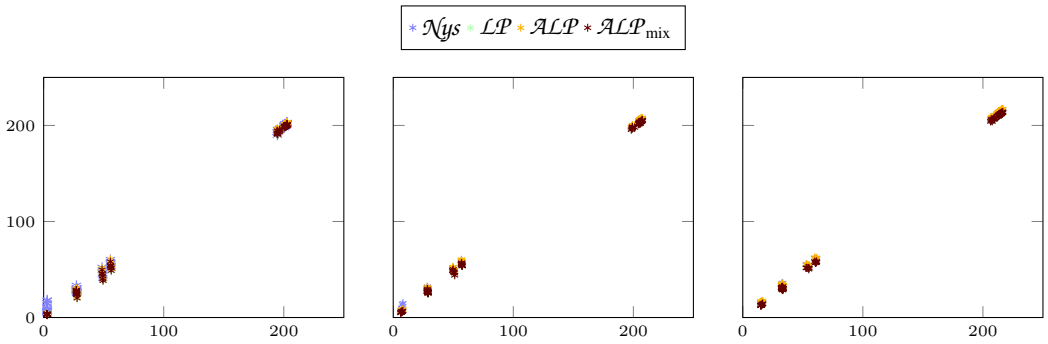
		100	250	500
Training		56.17%	51.01%	207.53%
Test	$\mathcal{N}ys$	59.08%	51.64%	207.47%
	$\mathcal{LP}$	56.22%	49.75%	205.87%
	$\mathcal{ALP}$	58.97%	52.33%	208.36%
	$\mathcal{ALP}_{\text{mix}}$	56.42%	49.54%	205.84%

differ markedly over the test subsample. This fact has to be taken into account when comparing the approximate embeddings computed for test patterns with their full sample counterparts.

These different distance values are presented in Table 5.4, that shows the median of the corresponding relative distances given as percentages when they are computed over the different training-test partitions. Because of the preceding discussion, we use the median as it is more robust than the mean in a case where we obtain $\mathbf{DM}_{\text{extd}}$ embeddings very different from $\mathbf{DM}_{\text{full}}$ ones.

We can observe how these relative Frobenius distances grow with the number of test points, as expected. And notice how, when we take 500 points out of the sample set for testing, the median of the relative Frobenius distance may be above 100%, which means that the training and test embeddings obtained are totally different from the ones computed with the full DM. This is possibly due to the large size of the test subset relative to the training one (recall that 500 points are about 25% of the sample set). In any case, we remark again that the train and test embeddings behave in the same way when compared in terms of the relative distances to their counterparts when an embedding is computed over the entire sample. In other words, the Nyström and LP embeddings result in test coordinates with a behavior very similar to that of the train coordinates.

In general, $\mathcal{LP}$ and $\mathcal{ALP}_{\text{mix}}$ present better reconstruction errors for all the test set sizes considered. As mentioned above, we cannot expect the "real" $\mathbf{DM}_{\text{full}}$ and extended $\mathbf{DM}_{\text{extd}}$ embeddings of the test subset to be close when this is not the case for the original embeddings computed by the direct eigenanalysis of the training patterns. In fact, a low train distance almost always correspond to a low test distance and the table exemplifies this, as test and train distances are rather similar in all cases. This effect can be checked also over Figure 5.5, where we present for the different test set sizes the relative Frobenius distances of the direct training subsample embeddings versus the relative distances over the extended testing subsample embeddings for the 100 experiments comparing the fourth models. Observe that the points essentially appear in the diagonal of the graph for all the embeddings, being more dispersed in the 100-size subsets and more concentrated in the 500-size.



(A) 100-test set (B) 250-test set (C) 500-test set

FIGURE 5.5: Relative Frobenius distances over the training subsample embedding versus the relative Frobenius distances over the testing subsample embedding for the 100 experiments done for each different test set size with the four out-of-sample models in consideration.

TABLE 5.5: A cost example of each out-of-sample method in time compression.

	Nys	*LP*	*ALP*	ALP_{mix}
100	7.56	838.76	161.00	306.37
250	16.59	688.32	148.15	273.99
500	28.16	540.47	127.49	220.73

At last we can compare the computational time required by each of these methods, which is shown in Table 5.5. As expected, the Nyström method is the fastest method, as it does not involve a training phase nor a parameter selection. To compare it against the second faster model, which is *ALP*, recall that, as mentioned in Section 5.2.2.1, in these experiments $m \gg l\bar{m}$, and because of this the main cost in the LP training without a CV phase (that coincides with the *ALP* training cost) is dominated by $O(mn^2)$ and in test by $O(mn)$. We can then say that the distance computation at the training phase takes the largest time. In the Nyström case the dominating part in the cost analysis grows as $O(mn)$. And, as in this case we have $\tilde{n}$ test patterns, the Nyström cost will grow as $O(\tilde{n}mn)$. This can give as an idea about how the relation between Nyström and *ALP* should be: Nyström should be around $n/\tilde{n}$ times faster than *ALP*. Following this estimation, Nyström should be around 20, 8 and 4 times faster than *ALP* for the 100, 250, and 500 test subset sizes respectively. In fact, this approximation is very close to the actual times obtained in Table 5.5.

Regarding to the other three models taken into consideration, the classic *LP* method is about 5 times costlier than the proposed *ALP*. This is due to the 5-fold CV used to decided the stopping criterion in the LP algorithm. The intermediate model proposed, the ALP_{mix}, is approximately twice slower than *ALP* but it is still much faster than *LP*. In Table 5.5 we can also observe that, according to the complexity analysis of Nyström and LP methods, as Nyström is dominated by $O(\tilde{n}mn)$, it grows with the size of the test subset. But in this concrete example, the LP-based

methods, as they are dominated by the training cost $O(MN^2)$ and in this case the training set size becomes smaller when we increase the number of examples for testing, the time performance for LP methods decrease with the rise of the testing set size.

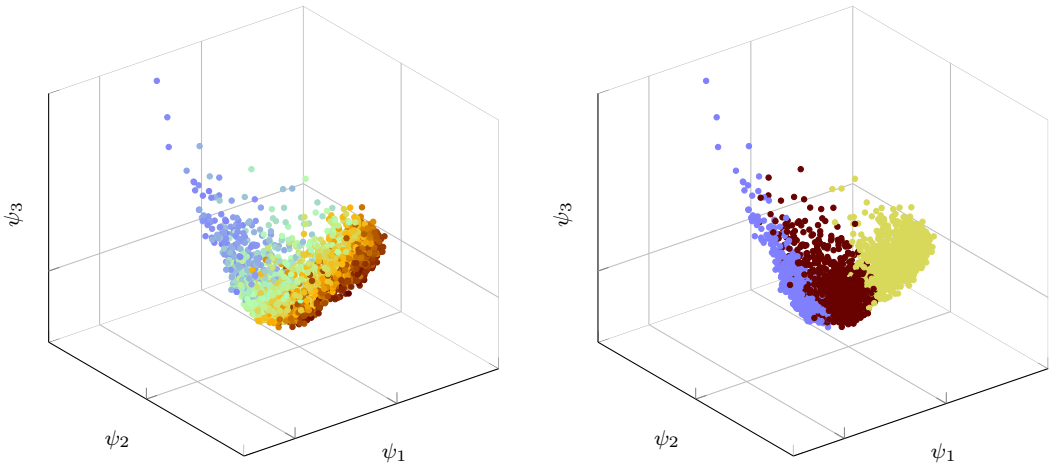
Along these lines we can conclude that $\mathcal{ALP}$ is a very good method for extending out-of-sample data coordinates, which improves considerably the computational cost versus the $\mathcal{LP}$ model (which uses CV for searching the best iteration number to stop), and in general produces better results or very similar ones when its $\mathcal{ALP}_{\text{mix}}$ modified version is used. The proposed model also outperforms the classical Nyström method in two of the three examples presented, and ties in the third one.

5.6 ALP for Solar Radiation Data

We finally consider a solar radiation problem where we have data from the 2013–2014 Solar Energy Prediction Contest organized by the American Meteorological Society (AMS) and hosted by the company Kaggle [Kaggle, 2014]. The ultimate goal in this competition is to predict the total daily incoming solar energy in 98 meteorological stations located in Oklahoma using Numerical Weather Predictions (NWP) forecasts of 15 variables for each one of the nodes of a grid that encompasses the state.

The complete input data of the contest are, on the one hand, an ensemble of eleven NWP forecasts from the NOAA/ESRL Global Ensemble Forecast System (GEFS) [GEFS, 2014], and, on the other, daily aggregated radiation readings from the 98 stations. For this experiment setting we have used only the main NWP forecasts ensemble that contains for every day five time steps (12 to 24 UTC-hours in three hour increments), giving for each one 15 variables related to radiation, temperature, precipitation, cloud clover and pressure. These forecasts are given in a grid of $16 \times 9 = 144$ points, having then each pattern a $144 \times 15 \times 5 = 10,800$-dimension where the spatial spread (144 points) dominates over the time (5) or variable (15) dimensions. These patterns are given for the years between 1994 and 2007, i.e., a total of $5,113$ days and, hence, patterns.

Our first goal is to compress for each day these daily $10,800$-dimensional spatial features. For this purpose, we have applied first the DM algorithm over the whole data set to obtain a low dimensional embedding. In this case, the parameter selection has been done as in the previous example (Section 5.5), selecting a σ value of 132.61 as the median of the Euclidean distances between points (the mean distance is now 138.03 and 48.73 the standard deviation) and we have fixed $\alpha = 1$ and $t = 1$. To decide on the embedding dimension, we have used a precision of $\delta = 0.1$ (i.e., we discard dimensions whose eigenvalues are smaller than 10% of the first one), reducing the initial $10,800$ dimension to just 3.



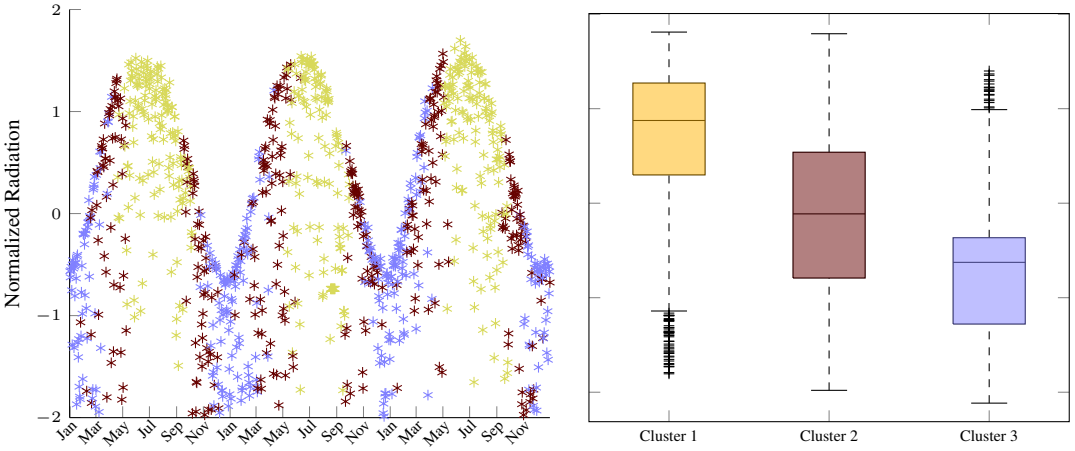
(A) DM Embedding colored by radiation. (B) κ-means clusters over DM coordinates.

FIGURE 5.6: Diffusion Coordinates, colored by the real incoming solar radiation average in the left hand image, and by the resulting k-means clustering over the embedding, with $k = 3$, in the right hand image.

The resulting embedding over the whole data set is shown in the left image of Figure 5.6, where its coordinates have been colored by the average over the 98 stations of their daily-aggregated incoming solar radiation. Observe that this measured radiation is not one of the NWP variables and, thus, is not included in the embedding. Even so, we can appreciate that the three-dimensional DM embedding captures quite clearly the average radiation in a band-like structure, with high radiation days (red and brown points) being clearly separated from low ones (blue and dark blue points). This band-like structure is approximately parallel to the first embedding axis, with a higher density of high radiation points (red and brown dots) to its right and a higher density of low radiation points (blue) to its left. Besides the fact that several of the NWP variables have only an indirect effect on radiation, we point out that radiation is perhaps one variable on which NWP models have a largest degree of uncertainty. Thus we cannot hope that the embedding captures radiation with a high precision, given that this is not the case in the starting NWP radiation forecasts.

To try to understand the DM embedding, we have built clusters over the embedding coordinates and checked their relationship with the measured radiation patterns. With this objective in mind, we have applied κ-means over the diffusion coordinates, with $\kappa = 3$, hoping to detect perhaps in the embedded data days with high radiation, days with intermediate radiation and days with low radiation. The resulting clusters are depicted in the right hand side of Figure 5.6 and show how clusters are defined mostly along the first DM dimension. Comparing both images in Figure 5.6 and taking into account the previous discussion, we could say that the DM clusters capture the high density areas of points with low, medium and high average radiation values.

To better visualize the possible meaning of these clusters, we have depicted the radiation time series colored by the cluster assigned to each day. This is shown in the left hand side of Figure 5.7



(A) Three years' radiation colored by cluster. (B) Radiation Boxplots.

FIGURE 5.7: Other visualization methods for the embedding clusters effect.

for the first three years in the sample, and we can see a structure of relatively wide green and blue vertical bands, corresponding approximately to summer and winter months respectively, and thinner intermediate brown vertical bands that approximately dominate spring and fall.

Moreover, in the right hand side of the figure we have drawn box plots for the distribution of average radiation in each cluster. We can see that the medians of the three clusters are well separated and the extreme radiation boxes overlap only at their respective outliers; the intermediate cluster has a higher overlapping but this is not incompatible with its assignment of time periods between summer and winter. Looking at these images we can conclude that DM achieves a high degree of spatial compression of the NWP data, with the diffusion coordinates capturing the regions with a higher density of low, medium and high radiation days and also representing the seasonality that is intrinsically present in radiation measurements.

In this example we are going to apply a two step ALP procedure to first extend DM embedding coordinates to test patterns and then to derive radiation predictions over these coordinates. In more detail, we begin by applying DM over the training sample to obtain the DM embedded features and build then a first ALP model $\mathcal{ALP}^F$ to extend the DM features to the test sample; next we build a second ALP $\mathcal{ALP}^R$ to approximate the training target radiations. Now, given a new NWP test pattern, we apply first $\mathcal{ALP}^F$ to obtain its extended DM features and, next, $\mathcal{ALP}^R$ over them to obtain the final radiation prediction. For all these experiments we will work with normalized data to 0 mean and a standard deviation 1.

First of all we are going to extend the embedding coordinates to the test subset, and study the performance of ALP compared to other out-of-sample methods just proven in Section 5.5 for extending diffusion coordinates to new, unseen patterns: the Nyström method called $\mathcal{Nys}$, the classical LP method named $\mathcal{LP}$ and the new proposed ALP. As done in the previous example, we consider here two variants of the ALP method: the proposed ALP method denoted as $\mathcal{ALP}^F$,

TABLE 5.6: Confusion matrices of the out-of-sample extensions for the solar dataset.

	P_1	P_2	P_3	$\sum$
R_1	303	0	0	303
R_2	23	325	0	348
R_3	0	11	433	444
$\sum$	326	336	433	1095

(A) $\mathcal{N}ys$ model. Accuracy: 96.89%

	P_1	P_2	P_3	$\sum$
R_1	303	0	0	303
R_2	10	338	0	348
R_3	0	3	441	444
$\sum$	313	341	441	1095

(B) $\mathcal{LP}$ model. Accuracy: 98.81%

	P_1	P_2	P_3	$\sum$
R_1	303	0	0	303
R_2	9	339	0	348
R_3	0	6	438	444
$\sum$	312	345	438	1095

(C) $\mathcal{ALP}^F$ model. Accuracy: 98.63%

	P_1	P_2	P_3	$\sum$
R_1	303	0	0	303
R_2	10	338	0	348
R_3	0	3	441	444
$\sum$	313	341	441	1095

(D) $\mathcal{ALP}^F_{\text{mix}}$ model. Accuracy: 98.81%

and the mixture between LP and ALP labeled as $\mathcal{ALP}^F_{\text{mix}}$. We are going to use the previously described NWP forecasts for the years 1994–2004 as the training set ($4,018$ patterns), and the years 2005, 2006 and 2007 for testing purposes ($1,095$ patterns), taking advantage of the natural ordering that makes possible in time dependent patterns to work with a single train-test split.

In these new experiments, we first compute an embedding over the training set, and we apply then the out-of-sample methods to extend the coordinates on the test points. We will also make use of the DM embedding previously computed over the entire sample, for comparing purposes as we have done with the time compression example. For evaluating the results obtained we use the same tools presented in Section 5.5: confusion matrices over the previously defined three clusters and relative Frobenius distances between the "true" and extended diffusion coordinates. The confusion matrices and the accuracy obtained for the classification problem between "real" and predicted cluster are shown in Table 5.6. As in the previous example, we can also see now high accuracies for all the methods, but $\mathcal{LP}$ and $\mathcal{ALP}^F_{\text{mix}}$ outperform the other methods. As explained before, it is logical that these two methods behave in the same way as the training and test models are built in the same manner except for the number of iterations that is decided differently. In this case they stop at different levels ($k = 10$ for $\mathcal{LP}$ while $k = 8$ for $\mathcal{ALP}^F_{\text{mix}}$) but both number of iterations is quite enough to just consider a very small neighborhood when constructing the kernel, making both results essentially equal.

To evaluate how similar is the reconstructed matrix to the one obtained computing directly DM over the whole set we define the relative Frobenius distance between both matrices. The results are presented in Table 5.7. Recall that, as before, this distance is presented for three different matrices. We observe that the distances for this example are lower than for the time compression one, giving a very good reconstruction. Even all the methods perform well and are almost equal,

TABLE 5.7: Median relative Frobenius distances between exact and approximated embedding coordinates computed for the solar radiation problem.

		$\|\|\|_F$
Train		13.45%
Test	$\mathcal{N}ys$	13.58%
	$\mathcal{LP}$	14.33%
	$\mathcal{ALP}^F$	14.32%
	$\mathcal{ALP}^F_{\text{mix}}$	14.33%

TABLE 5.8: Cost in seconds of each out-of-sample method for the solar radiation example.

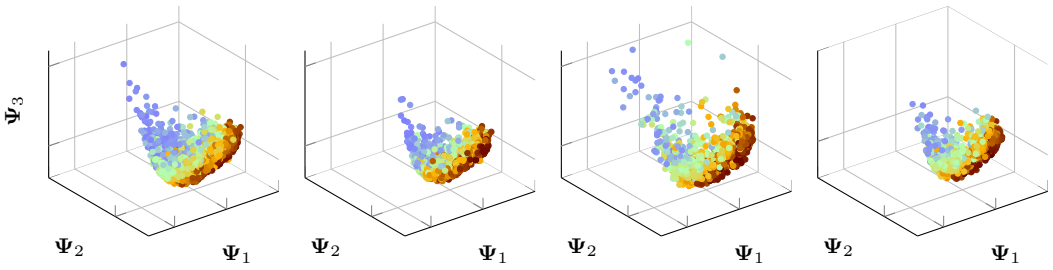
	$\mathcal{N}ys$	$\mathcal{LP}$	$\mathcal{ALP}^F$	$\mathcal{ALP}^F_{\text{mix}}$
Time (s)	125.00	2,641.16	607.01	1,107.17

the Nyström method is slightly better, if we consider the objective to recover the 'complete" DM matrix.

Note that the results obtained over this solar data example are more robust. One reason is probably that we have more data and the training and test sets constructed are better representatives of the problem to solve (we counted with seven complete years for training and three for test). Another, and perhaps more important reason is that here we can expect a greater similarity between the training and test subset patterns, as radiation (and to some extent, weather itself) is clearly a seasonally periodic phenomenon where patterns in the testing years cannot be extremely different from patterns in the previous training years. This similarity makes it somehow easier the extension of the embeddings.

We can compare the computational time required by each of these methods in Table 5.8. Again, it can be appreciated that the Nyström method is the fastest method, followed by the $\mathcal{ALP}^F$ model. The relation between these two models will be again determined by the quantity $n/\tilde{n}$, which in this case means that Nyström is about 4.5 times faster than $\mathcal{ALP}$; looking at the table, this is a good approximation for the different times needed by each model. The $\mathcal{ALP}^F_{\text{mix}}$ model is about twice more expensive than $\mathcal{ALP}^F$ but still less than $\mathcal{LP}$, as expected. The $\mathcal{LP}$ is the costliest model, being about 5 times slower than the proposed $\mathcal{ALP}^F$ method, as it computes a 5-fold cross validation when looking for the best number of iterations to stop the model.

In any case, we can also conclude here that Nyström, LP and ALP are good methods for extending diffusion coordinates. As mentioned, Nyström's method is much faster but in this case it seems that if clustering is sought, the LP models result in better test accuracies that, in the case of ALP, may compensate for its costlier training to result in a better choice.



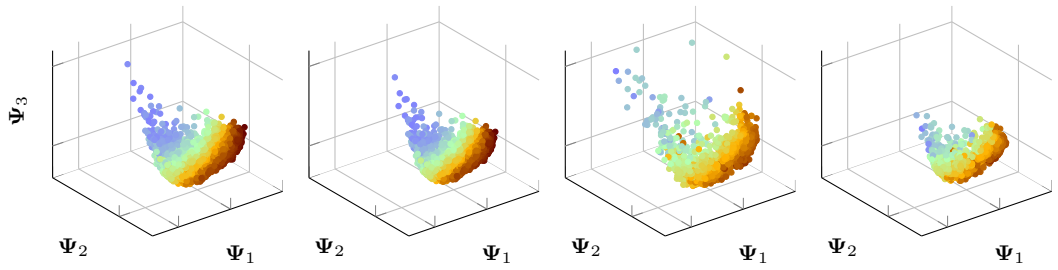
(A) DM Train. (B) Full DM Train. (C) $\mathcal{ALP}^R$ Test. (D) Full DM Test.

FIGURE 5.8: Training and test results when the DM embedding is computed only on the training sample and $\mathcal{ALP}^F$ is used to extend DM to the test sample compared against the full DM; colored by real averaged radiation.

After extending the embedding coordinates we are ready to apply $\mathcal{ALP}^R$, which recall is the radiation prediction ALP model, over the embedded coordinates, getting an approximation to the average of the total daily incoming solar radiation in the 98 Oklahoma's meteorological stations. For this purpose, we used the extended variables obtained by $\mathcal{ALP}^F$ as we have just seen that it is a computationally efficient method that yields good results. The quality of the embedding obtained can be also seen in Figure 5.8 where we plot the different embeddings (the one obtained applying DM to the whole data set and the one where the test subset has been obtained by extension of the training one) colored by the averaged radiation of the 98 stations. It can be appreciated that the DM embeddings over training and test subsets obtained by both methods are very similar and that the target colors present the same order. This shows that when we apply $\mathcal{ALP}^F$ to compute the DM coordinates of new test sample points we get an embedding quite close to the ideal one obtained jointly over the train and test patterns.

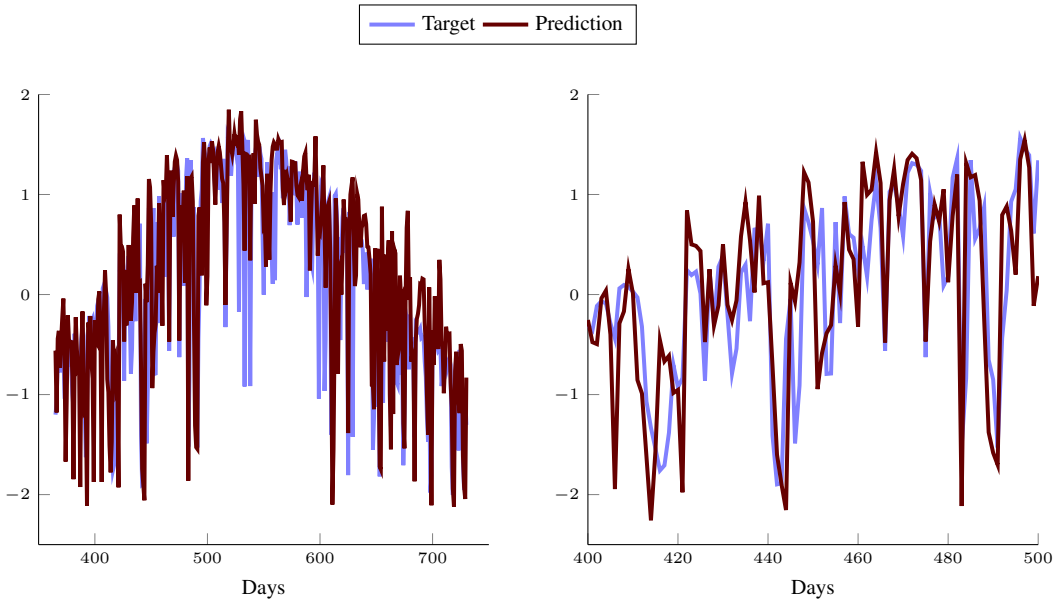
The same embeddings can be depicted but colored by the $\mathcal{ALP}^R$ predicted radiation. The result is shown in Figure 5.9, where, comparing with Figure 5.8, we can observe that the radiation values have been smoothed across color bands and that the general radiation trend is captured approximately along the second DM feature even if not every detail is modeled (recall that measured radiation is the target value and, thus, is not included in the DM transformation). This behavior can be observed for both, training and test points, corroborating that the $\mathcal{ALP}^R$ method makes a good extension of the target function for new points. Comparing the $\mathcal{ALP}^F$ method that extends the embedding coordinates with the full one, we can again appreciate how the prediction colors seem to be more or less the same over both training and test subsets. Comparing with Figure 5.8 we can see that the two DM embeddings are very similar and that the target and prediction colors seem to be more or less the same.

To close this section and for a better understanding of how ALP performs over a real example, we present in Figure 5.10, left, the real averaged radiation in blue and the two step ALP prediction in red for the second test year. It can be seen that ALP captures well radiation's seasonality. In the right plot we zoom in and it can also be appreciated how $\mathcal{ALP}^R$ tracks the radiation variations.



(A) DM Train. (B) Full DM Train. (C) $\mathcal{ALP}^R$ Test. (D) Full DM Test.

FIGURE 5.9: Training and test results when the DM embedding is computed only on the training sample and $\mathcal{ALP}^F$ is used to extend DM to the test sample compared against the full DM; colored by $\mathcal{ALP}^R$ radiation prediction.



(A) (B)

FIGURE 5.10: Prediction of the daily incoming solar energy over the second test year, and in a zoom over 100 days.

Even if not every peak is caught (the winning model in the Kaggle competition followed a different approach), $\mathcal{ALP}^R$ yields a reasonably good approximation to actual radiation neither requiring particular parameter choices nor an expert knowledge about the problem we wanted to address.

Figure 5.11 shows the evolution of standard LP training error, its associated LOOCV error and the LOOCV estimation given by ALP. It illustrates the robustness of the ALP model against overfitting. Again, the ALP model requires here the same number of 15 iterations suggested by applying full LOOCV to standard LP.

Finally, Figure 5.12 shows for an embedded test point $\mathbf{x}$, plotted as a black dot, the ALP evolution in terms of the "influence" of sample points on $\mathbf{x}$, that is the value of the kernel $\mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x})$,

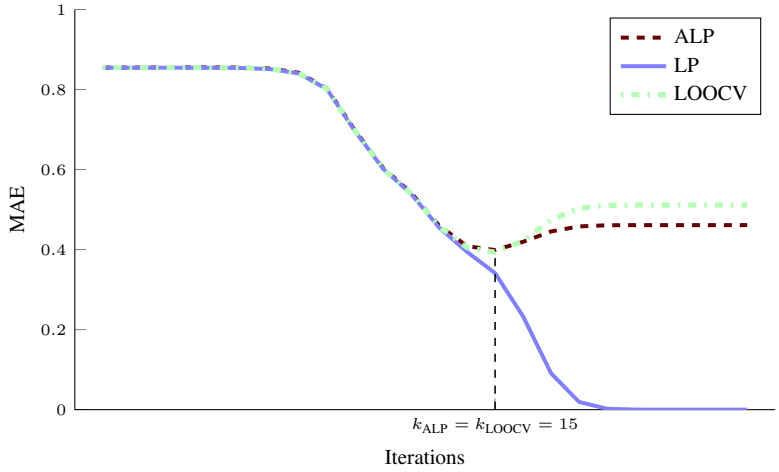


FIGURE 5.11: Training errors for the solar example of ALP, standard LP and its associated LOOCV.

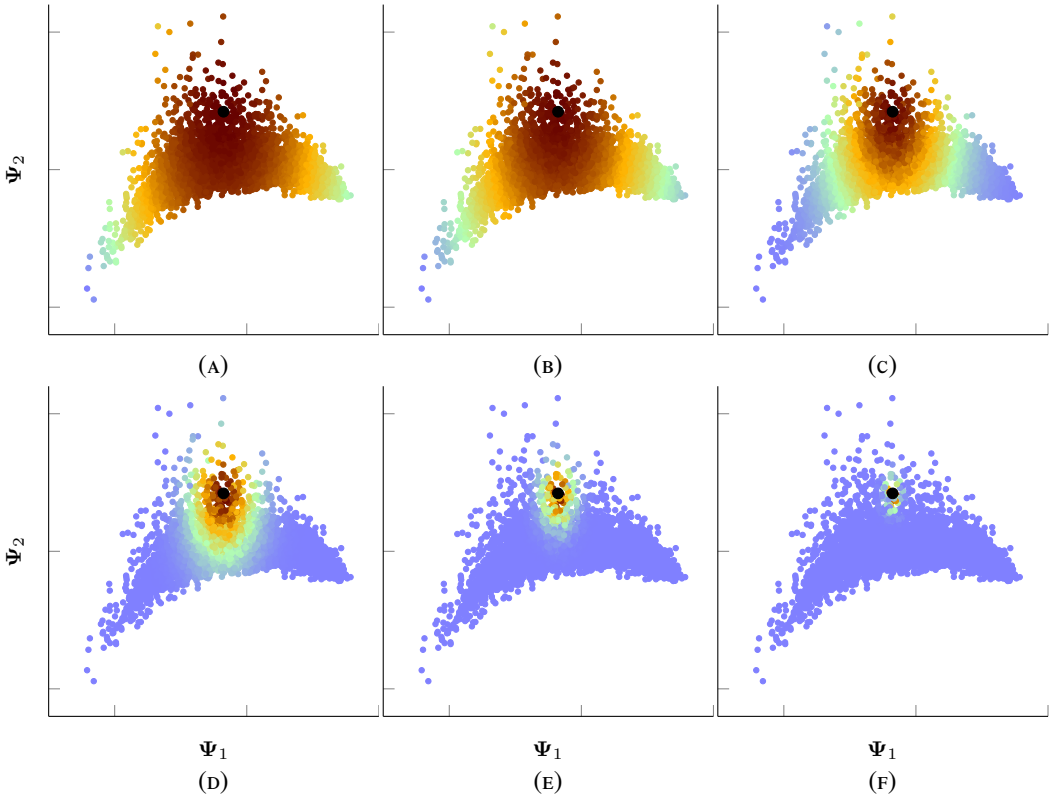


FIGURE 5.12: Evolution of the neighborhood of a test point over the embedded training points.

where red points indicate a stronger influence and blue ones a smaller one. Note that, as σ gets smaller, the number of high influence red points also decreases sharply and so does the possibility of overfitting.Thence, ALP does not overfit the data and does not require any parameterization or expert knowledge about the problem, while still achieving a good test error. Moreover, it adds no extra cost compared to other classical neighbor-based interpolation methods.

Chapter 6

Conclusions

6.1 Discussion and Conclusions

This thesis has presented a systematic compendium of manifold learning methods, especially focused on diffusion methods.

First of all, some classical spectral dimensionality reduction methods have been thoroughly reviewed, comparing and connecting them, and finally unifying them under the more general framework of Diffusion Maps (DM). In particular, we have shown that DM is an useful technique for dimensionality reduction, since it preserves quantities of interest such as local mutual distances. Moreover, the resultant features of DM can help to gain insight and understanding about the underlying phenomenon that generates the data and also about its meaningful structures. As additional issues, we have discussed how to take into account the sample density influence, we have included a brief guide for parameter selection and we have shown how DM can be used as a natural technique for homogenizing attributes.

We have illustrated that DM is a good method for data compression and visualization. More specifically, an analysis of the meteorological data for the Iberian Peninsula has been done, considering both a spatial and time compression of the information. In this context, DM is able to capture well the underlying structure, providing a very low dimensionality description. Furthermore, DM dimensionality reduction and clustering are effective tools for building local models in wind energy forecasting problems.

On the other side, Anisotropic Diffusion (AD) is another recent diffusion method that has been studied in this thesis. This method assumes that the sample is generated by a nonlinear function acting over some latent features that follow an Itô process, and it tries to invert this function in a kind of nonlinear component analysis. The main difference with respect to DM is that the Euclidean distance used to define the Gaussian kernel is replaced by a local Mahalanobis distance. We have taken advantage of this idea through a model based on this distance but which is built directly over the observable space, without needing explicitly the features of the embedding, and thus avoiding the costly eigenanalysis. In particular, the proposed algorithm consists in defining a κ-Nearest Neighbors (κ-NN) regression model using the local Mahalanobis distances, which can be shown to approximate the Euclidean distances between the latent variables.

This algorithm has been applied to two concrete problems: Computed Tomography (CT) scan images location in the human body and wind ramps detection. We have seen that it has a good performance over both applications. Moreover, it should be remarked that wind ramps detection is a very new but important problem, without reference results yet; and in this thesis, we have presented a methodology for solving it.

A big challenge for DM is the out-of-sample embedding coordinates extension. We have seen that for embedding new, unseen points without repeating the whole DM process we should approximate their corresponding eigenvectors. In the literature, the Nyström formula and the Laplacian Pyramids (LP) method are two of the most used approaches. A new variant of LP, namely Auto-adaptative Laplacian Pyramids (ALP), is proposed in this thesis. It has been shown that this new algorithm is a good method for extending out-of-sample data coordinates with the advantages that it does not overfit the data and, moreover, it does not essentially require any parameterization or expert knowledge about the problem. The behavior of this method has been exemplified over a synthetic example and real meteorological data set based on wind and solar radiation.

These out-of-sample methods have been compared using some proposed measures that quantify how well these algorithms are extending the coordinates. More concretely, we have presented a meteorological wind data and a solar radiation data examples, where we have seen how the ALP algorithm is faster than other LP methods and obtain equal or better results than the other out-of-sample methods.

In summary, it can be said that in this thesis a compact, self-contained framework for diffusion methods has been presented, offering and comparing the different existent algorithms and its theoretical background. Moreover, two different algorithms have been proposed, dealing with two of the most important challenges in diffusion methods: the costly embedding computation and the extension of the methods for out-of-sample data. Also, an application contribution has been done, showing how both the state-of-the-art methods and the proposed ones can be

successfully applied to real problems, more concretely in renewable energies and medical image related problems.

6.2 Further Work

We discuss next some ideas and venues to extend the work presented here.

A first area of interest is that of defining better procedures to determine the various parameters present in DM and AD. For instance, when using a Gaussian kernel to build a similarity matrix, the selection of the best width parameter is an important and problem dependent issue. In this work, we have proposed to choose it in terms of appropriate percentiles of the sample distances but at the end the percentile finally chosen depends on an analysis of the problem at hand, but it should have a more systematic nature. The same happens with the selection of the most appropriate embedding dimension, where perhaps more attention should be paid on the difference between the Euclidean distance in the embedded space and the diffusion distance in the sample's manifold.

With respect to the AD-based model proposed, it can be redefined according to the ideas exposed in Szlam et al. [2008], adding an iterative refinement of the κ-NN prediction and focusing in obtaining in each step a better local covariance matrix so the predictions become more accurate. Also, different kernels from the Gaussian one should be explored; for example, it could be interesting to study what happens when we work with kernels that just take into account points in local neighborhoods appropriately defined.

Turning our attention to the clustering techniques, it would be highly desirable that the number of clusters κ could be selected automatically or at least problem-independently when applying κ-means. Notice that in our experiments the number of clusters has been selected beforehand either according to previous knowledge about the problem to be solved or taking into account considerations such as the convenience for visualization purposes. For example, in Hamerly and Elkan [2003] a method that determines in an automatic way the best κ for each problem, assuming Gaussianity in the clustered data, is proposed and it would be desirable to analyze it and other similar methods in more detail.

In this thesis we have shown that the DM embedding procedure can be adequately extended to new sample points either using the Nyström formula or our ALP proposal. To have effective extension procedures is crucial if one has in mind big data applications, but a more pressing issue is then to have good subsampling techniques that enable us to work with lower dimensional similarity matrices that still capture the relevant sample structure. We are currently working on this topic, and even it has not been included in this thesis, we have proposed a subsampling method that uses the cluster centroids obtained by applying kernel κ-means. It will be important

to study other similar techniques that improve the result, and, in this line, it will be also desirable to invest some effort on defining better evaluation techniques.

As far as applications are concerned, we intend to follow our work on the analysis of meteorological data, either by themselves or applied to renewable energy prediction. We believe that the very high data dimensionality makes embedding methods such as DM or AD to be important tools in this field. Wind ramp detection is an example of an important problem where there are not reference results yet and in which manifold learning and diffusion methods can have an impact. An area we are currently working on is to improve the thesis' results using richer meteorological patterns than those considered here or using more sophisticated wind energy forecasting methods as baseline models.

Another important problem is the detection and analysis of singular days where actual meteorological behavior markedly differs from what it was to be expected from Numerical Weather Predictions (NWP) forecasts. Finally, we believe that solar energy is set to achieve a development as extensive as the current one of wind energy. Radiation, the main meteorological variable in solar energy, is a much more seasonal phenomenon than wind, and while at individual farms clouds and other factors can produce fast and marked variations in energy output, the behavior of solar energy over larger areas is more stable. These considerations suggest the exploration of solar energy models based on the past behavior of previous days for which NWP predictions were similar to that for the current one, in the same way as done here for wind ramp detection. We think that DM or, perhaps better, AD approaches can be very well suited to these tasks and we are currently working on these and other related issues.

Capítulo 6

Conclusiones

6.1 Discusión y Conclusiones

En esta tesis se ha presentado un compendio sistemático de métodos de aprendizaje de subvariedades, centrado especialmente en métodos de difusión.

En primer lugar, algunos métodos espectrales clásicos de reducción de dimensión han sido cuidadosamente revisados, comparándolos y conectándolos entre sí para, al final, unificarlos bajo el marco general de *Diffusion Maps (DM)*. En particular, se ha visto que DM es una técnica muy útil para reducir la dimensión de un conjunto, ya que es capaz de mantener cantidades interesantes como las distancias mutuas locales. Más aún, las variables resultantes de aplicar DM pueden ayudar a ganar conocimiento y entendimiento sobre el fenómeno subyacente que genera los datos y también sobre sus estructuras más significativas. Como aspectos adicionales, se ha discutido cómo tener en cuenta la influencia de la densidad de la muestra, se ha incluido una breve guía sobre cómo seleccionar los parámetros y se ha mostrado cómo DM se puede usar como técnica para homogeneizar atributos.

Se ha ilustrado cómo DM es un buen método para comprimir y visualizar datos. Más específicamente, se ha realizado un análisis de datos meteorológicos en la península Ibérica, considerando tanto una compresión espacial como temporal de la información. En este contexto, DM es capaz de capturar bien la estructura subyacente, proporcionando una descripción de los datos en muy baja dimensión. Más aún, se ha visto como la reducción de dimensión mediante DM combinada

con técnicas de *clustering* son herramientas muy efectivas para construir modelos locales en problemas de predicción de energía eólica.

Por otro lado, *Anisotropic Diffusion (AD)* es otro método reciente de difusión que ha sido estudiado en esta tesis. Este método asume que la muestra ha sido generada por una función no lineal que actúa sobre ciertas variables latentes que siguen un proceso de Itô, y trata de invertir dicha función en una especie de análisis de componentes no lineal. La principal diferencia respecto a DM es que las distancias Euclídeas usadas para definir el núcleo Gaussiano se reemplazan aquí por una distancia local de Mahalanobis. Hemos sacado partido a esta idea mediante la definición de un modelo basado en esta distancia, pero que se construye directamente sobre el espacio observable, sin la necesidad de embeber explícitamente las características y ahorrándonos, por tanto, el costoso análisis de autovalores. En particular, el método propuesto consiste en definir un modelo de vecinos próximos (κ-NN) de regresión usando distancias de Mahalanobis, las cuales se puede demostrar que aproximan las distancias Euclídeas entre variables latentes.

Este algoritmo se ha aplicado en dos problemas concretos: el etiquetado de tomografías y la detección de rampas. Se ha comprobado que este método ofrece buenos resultados sobre ambos problemas. De hecho, se debe recalcar que el problema de detección de rampas es un problema muy nuevo, pero muy importante, sin resultados de referencia todavía, y en esta tesis se ha presentado una metodología para resolverlo.

Un gran reto para los DM es la extensión de las coordenadas del *embedding* a puntos de fuera de la muestra. Hemos visto que para embeber puntos nuevos sin repetir todo el proceso de DM hay que aproximar los autovectores correspondientes. En la literatura, dos de las aproximaciones principales para esto son la fórmula de Nyström y el método de *Laplacian Pyramids (LP)*. En esta tesis se ha propuesto una variante de LP, llamada *Auto-adaptative Laplacian Pyramids (ALP)*. Este nuevo algoritmo es un buen método para extender puntos de fuera de la muestra con la ventaja de que no sobreajusta los datos y, además, no requiere ninguna parametrización o conocimiento experto sobre el problema. Se ha ejemplificado el comportamiento de este método sobre un problema sintético y sobre datos meteorológicos reales de viento y radiación solar.

Se han propuesto también algunas medidas que permiten comparar estos métodos de extensión de autovectores, cuantificando cómo de bien extiende las coordenadas cada algoritmo. En concreto, se han presentado ejemplos con datos meteorológicos relacionados con viento y con radiación solar, donde se ha mostrado como el algoritmo ALP es más rápido que LP, obteniendo resultados equivalentes o incluso mejores.

En resumen, se puede decir que en esta tesis se ha presentado un marco compacto y autocontenido para métodos de difusión, detallando y comparando los distintos algoritmos existentes y su contexto teórico. Además, se han propuesto dos algoritmos diferentes, centrados en los dos retos más importante de los métodos de difusión: el costoso cálculo del *embedding* y la extensión

de los métodos a datos de fuera de la muestra. Asimismo, se ha hecho una contribución de aplicación, ilustrando que tanto los métodos del estado del arte como los propuestos pueden aplicarse de forma exitosa a problemas reales, concretamente a problemas relacionados con energías renovables y con imágenes médicas.

6.2 Trabajo Futuro

A continuación se discuten algunas ideas y escenarios para extender el trabajo aquí presentado.

Una primera idea de interés es definir mejores procedimientos para determinar los distintos parámetros utilizados en DM y AD. Por ejemplo, cuando utilizamos el núcleo Gaussiano para construir la matriz de similitud, la elección del mejor valor para el parámetro de anchura del núcleo es una cuestión importante y dependiente del problema. En este trabajo se propone elegirlo en términos de un percentil de las distancias en la muestra pero, al final, el percentil elegido depende de un análisis del problema a resolver, y sin embargo sería deseable que tuviese una naturaleza más sistemática. Pasa lo mismo con el tema de la elección de la dimensión más adecuada para el *embedding*, donde se podría quizás prestar más atención a las diferencias existentes entre la distancia Euclídea en el espacio embebido y la distancia de difusión en la subvariedad de la muestra.

Respecto al modelo propuesto basado en AD, se podría redefinir utilizando las ideas propuestas en Szlam et al. [2008], añadiendo un refinamiento iterativo de las predicciones de κ-NN, con vistas a obtener en cada paso una matriz local de covarianza mejor y en consecuencia, unas predicciones más precisas. Además, se podrían explorar otros métodos de núcleo distintos al Gaussiano, por ejemplo sería interesante estudiar qué ocurre cuando se trabaja con núcleos que sólo tienen en cuenta puntos en un vecindario local definido adecuadamente.

Volviendo nuestra atención a las técnicas de *clustering*, sería muy deseable que el número κ de *clusters* que se buscan al aplicar *κ-means* se pudiese seleccionar de forma automática o, al menos, de forma independiente del problema. Nótese que en nuestros experimentos el número de *clusters* ha sido seleccionado a priori o bien partiendo de conocimiento previo del problema, o bien teniendo en cuenta otras consideraciones como una apropiada visualización del resultado. Por ejemplo en Hamerly and Elkan [2003] se propone un método que determina de forma automática el mejor κ para cada problema, asumiendo gaussianidad en los datos clasificados. Sería deseable analizar éste y otros métodos parecidos en más detalle así como proponer un método más general para determinar el valor de este parámetro.

En esta tesis se ha visto que el mecanismo de *embedding* de DM se puede extender adecuadamente a nuevos puntos en la muestra usando o bien la fórmula de Nyström o bien nuestra propuesta del algoritmo ALP. Contar con procedimientos de extensión efectivos es crucial si

uno tiene aplicaciones *big data* en mente, y un tema urgente a tratar en este sentido es el de contar con técnicas de submuestreo que permitan trabajar con matrices de similitud de menor dimensión pero que mantengan la estructura relevante de la muestra. Estamos trabajando actualmente en este tema y, aunque no se ha incluido en esta tesis, hemos propuesto una nueva técnica de submuestreo que utiliza los centroides de los *clusters* obtenidos al aplicar *kernel* κ*-means*. Sería importante estudiar otras técnicas similares que mejoren el resultado obtenido, y en esta línea, sería deseable invertir además algún esfuerzo en la definición de mejores técnicas de evaluación de este tipo de métodos.

En lo que concierne a las aplicaciones, se pretende continuar el trabajo iniciado en el análisis de datos meteorológicos, siendo interesantes bien por si mismos, o bien aplicados a la predicción de energías renovables. Creemos que los datos de muy alta dimensión hacen de los métodos de *embedding*, como DM o AD, herramientas importantes en este campo. La detección de rampas es un ejemplo de problema importante en el que no hay todavía resultados de referencia y donde el aprendizaje de subvariedades y los métodos de difusión pueden tener un cierto impacto. Un área en la que estamos trabajando actualmente es la mejora de los resultados presentados en esta tesis, utilizando patrones meteorológicos más ricos que los aquí considerados o usando métodos de predicción de energía eólica más sofisticados como modelos de base.

Otro problema importante es la detección y análisis de días singulares en los que el comportamiento meteorológico difiere notablemente de lo esperado según las predicciones numéricas. Finalmente, creemos que la energía solar es un campo donde obtener un desarrollo tan exhaustivo como el actual en energía eólica. La radiación, que es la principal variable meteorológica en el problema de energía solar, es un fenómeno mucho más estacional que el viento, y mientras que a nivel de granja las nubes y otros factores pueden producir rápidas y marcadas fluctuaciones en la producción, el comportamiento de la energía solar sobre grandes áreas es mucho más estable. Estas consideraciones sugieren que la exploración de modelos de energía solar basados en los comportamientos pasados de días previos con predicciones numéricas similares al dato en consideración, de una manera parecida a lo que se ha hecho para detectar rampas de viento. Creemos que las aproximaciones de DM o incluso mejor, de AD, pueden adaptarse muy bien a esta tarea y estamos actualmente trabajando en estas y otras cuestiones similares.

Appendix A

Relation of Publications

This appendix shows the list of publications done while this thesis was carrying out. The first list presents the publications directly related with this work, and the second one presents some collaboration papers done in other machine learning areas.

Diffusion Methods

The spectral dimensionality reduction methods reviewed in Chapter 2 were firstly presented in my Master's thesis:

Advanced Methods for Dimensionality Reduction and Clustering [Fernández, 2010]
Master's thesis.

In this work a study of the state-of-the-art in advanced dimensionality reduction and clustering techniques was presented. It was mainly focused on the explanation of Laplacian Eigenmaps (LE) and Spectral Clustering (SC) and on the detailed review of the Nyström formula as an extension method for embedding coordinates of new points.

The following two publications are the basis of Chapter 3:

Diffusion Maps for the Description of Meteorological Data [Fernández et al., 2012b]
Int. Conf. on Hybrid Artificial Intelligent Systems (HAIS 2012).

In this work Diffusion Maps (DM) was presented and applied to obtain a time and spatial compression of numerical weather forecasts, showing how this method is capable to

greatly reduce the initial dimension while still capturing relevant information in the original data.

Diffusion Maps and Local Models for Wind Power Prediction [Fernández et al., 2012a]
Int. Conf. on Artificial Neural Networks (ICANN 2012).

The construction of local models for wind energy forecasting is an attractive option due to the high variance nature of this data. In this work, DM has been used for defining clusters that will be useful for a cluster-specific model determination.

Chapter 4 is principally based on the three following papers:

Diffusion Maps for Wind Power Ramp Detection [Fernández et al., 2013a]
Int. Work Conf. on Artificial Neural Networks (IWANN 2013).

The prediction and management of wind power ramps is a crucial issue for system operators and wind farm managers, but it is still far from being solve. This paper proposed a framework to address it as a classification problem working with delay vectors of the wind power time series and applying a κ-Nearest Neighbors (κ-NN) search with metrics derived from Anisotropic Diffusion (AD) methods.

Local Anisotropic Diffusion Detection of Wind Ramps [Fernández et al., 2013b]
Workshop on Machine Learning for Sustainability of the NIPS Conference (NIPS 2013).

This work continues the previous one, proposing a refined prediction based on scores depending on the ramp presence or not in similar patterns in the past, making richer patterns and evaluating results under a ROC curves point of view.

Diffusion Methods for Location Prediction in CT Scan Images [Fernández et al., 2014b]
Medical Image Analysis.

The purpose of this study is to introduce diffusion methods as a tool to label CT scan images according to their position in the human body. A comparative study of different methods based on a κ-NN search is carried out and we propose a new, simple and efficient way of applying anisotropic diffusion techniques that is able to give better location forecasts than methods that can be considered the current state-of-the-art in this field.

The following works have not been published yet, but they present the main foundations of Chapter 5:

DM for DR and Visualization of Meteorological Data [Fernández et al., 2014a]
Submitted to Neurocomputing (HAIS 2012 Special Issue).

In this work we gave a self-contained review of DM and discussed two methods to compute its embedding coordinates to new out-of-sample data. We have applied them on two meteorological data problems that involve respectively time and spatial compression of numerical weather forecasts.

ALP for High-dimensional Data Analysis [Fernández et al., 2014]

Available at http://arxiv.org/abs/1311.6594.

In this paper we propose the Auto-adaptative Laplacian Pyramids (ALP) algorithm, an extension of the standard Laplacian Pyramids (LP) that incorporates a modified Leave One Out CV (LOOCV) procedure that avoids the large cost of standard LOOCV and offers the advantages of automatically selecting the stopping criterion (thus it does not need parameterization), not overfitting the training set and not adding extra cost compared to other classical interpolation methods.

Further Diffusion Methods Research

Kernel K-Means Low Rank Approximation for SC and DM [Alaíz et al., 2014]

Int. Conf. on Intelligent Data Engineering and Automated Learning (IDEAL2014).

SC and DM require the eigenanalysis of a sample's graph Laplacian, something very costly for moderately sized samples and prohibitive for very large ones. In this work it is proposed to build a low rank approximation using essentially the centroids obtained applying kernel κ-means over the similarity matrix.

Other Lines of Research

The following publications present some collaboration works done in other machine learning areas not directly related with this thesis:

High Wind and Energy Specific Models for Global Production Forecast [Alaíz et al., 2009]

European Wind Energy Conference (EWEC 2009).

High and low production regimes are in principle different enough as to warrant the use of regime–specific models, that in this work have been defined via some wind/production thresholds.

A Customer Management System for the Spanish Transport System Operator [Díaz et al., 2009]

Spanish-Portuguese Conf. on Electrical Engineering (CHLIE 2009).

This paper presents the software that has been designed and developed for the Spanish Customer Management System in the context of the Active Demand Management (GAD) project, whose main goal was to optimize the use of electricity and, therefore, the cost associated with that usage.

Support Vector Forecasting of Solar Radiation Values [Gala et al., 2013]

Int. Conf. on Hybrid Artificial Intelligent Systems (HAIS 2013).

The increasing importance of solar energy has made the accurate forecasting of radiation

an important issue, and machine learning tools can be useful to solve it. In this work Support Vector Machines (SVM) are applied to downscale and improve 3-hour accumulated radiation forecasts for two locations in Spain combined with a clear sky curve for disaggregation purposes.

Machine Learning Prediction of Global Photovoltaic Energy in Spain [Gala et al., 2014b]

Int. Conf. on Renewable Energies and Power Quality (ICREPQ 2014).

In this work we will explore the application of Support Vector Regression (SVR) to forecast the daily photovoltaic generation of Spain.

Hybrid Machine Learning Forecasting of Solar Radiation Values [Gala et al., 2014a]

Submitted to Neurocomputing (Special Issue HAIS 2013).

This work is an extension of the study presented in the HAIS 2013 conference, where Support Vector Regression (SVR), Gradient Boosted Regression (GBR), Random Forest Regression (RFR) and hybrid methods where applied to downscale and improve 3-hour accumulated radiation forecasts for seven locations in Spain.

BIBLIOGRAPHY

Aizenbud, Y., Bermanis, A., and Averbuch, A. (2014). PCA-based out-of-sample extension for dimensionality reduction. Submitted. http://www.cs.tau.ac.il/~amir1/PS/pbe.pdf. Cited on page 126.

Alaíz, C., Barbero, A., Fernández, A., and Dorronsoro, J. (2009). High wind and energy specific models for global production forecast. In *Proceedings of the European Wind Energy Conference and Exhibition - EWEC 2009*. EWEA. Cited on pages 70 and 163.

Alaíz, C., Fernández, A., Gala, Y., and Dorronsoro, J. (2014). Kernel k-means low rank approximation for spectral clustering and diffusion maps. In *International Conference on Intelligent Data Engineering and Automated Learning - IDEAL 2014*, Lecture Notes in Computer Science, Heidelberg, Germany. Springer Berlin Heidelberg. Cited on page 163.

Aldous, D. and Fill, J. (2002). Reversible markov chains and random walks on graphs. Unfinished monograph, recompiled 2014, available at http://www.stat.berkeley.edu/~aldous/RWG/book.html. Cited on pages 48 and 51.

Alexander, C. (2011). Wind ramp prediction. Master's thesis, Oregon State University. Cited on page 103.

Ball, G. and Hall, D. (1965). ISODATA, a novel method of data analysis and pattern classification. Technical report, DTIC Document. Cited on page 4.

Barbero, A., López, J., and Dorronsoro, J. (2008). Kernel methods for wide area wind power forecasting. In *Proceedings of the European Wind Energy Conference and Exhibition - EWEC 2008*, Brussels, Belgium. EWEA. Cited on page 77.

Barbour, P., Casey, S., and Walke, S. (2010). Evaluation of BPA vendors wind plant "Wind Ramp Event" tracking system. Technical Report BPA 2009-FR-PUBLIC, Energy Resources Research Laboratory, Oregon State University, Corvallis, OR 97331. Cited on page 104.

Belabbas, M. and Wolfe, P. (2009). Spectral methods in machine learning and new strategies for very large datasets. *Proceedings of the National Academy of Sciences of the United States of America*, 106(2):369–374. Cited on page 125.

Belkin, M. and Niyogi, P. (2001). Laplacian Eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 13 - NIPS 2001*, volume 14, pages 585–591. Cited on page 43.

Belkin, M. and Nyogi, P. (2003). Laplacian Eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396. Cited on pages 9, 14, 15, 16, 18, 19, 21, 24 and 43.

Bengio, Y., Delalleau, O., Roux, N. L., Paiement, J., Vincent, P., and Ouimet, M. (2006). *Spectral dimensionality reduction*. Springer. Cited on pages 3 and 9.

Bengio, Y., Paiement, J.-F., and Vincent, P. (2003). Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *Advances in Neural Information Processing Systems 15 - NIPS 2003*, pages 177–184. MIT Press. Cited on pages 125 and 127.

Bhat, Y. S. and Arnold, G. (2007). Diffusion Maps and radar data analysis. In *Proceedings SPIE*, volume 6568. Cited on page 56.

Biernacki, C., Celeux, G., and Govaert, G. (2000). Assessing a mixture model for clustering with the Integrated Completed Likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):719–725. Cited on page 57.

Bishop, C. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA. Cited on pages 4, 5, 7 and 43.

Bossavy, A., Girard, R., and Kariniotakis, G. (2013a). Forecasting ramps of wind power production with numerical weather prediction ensembles. *Wind Energy*, 16(1):51–63. Cited on pages 103, 104 and 110.

Bossavy, A., Girard, R., and Kariniotakis, G. (2013b). A novel methodology for comparison of different wind power ramp characterization approaches. In *European Wind Energy Association Annual Event - EWEA 2013*, Vienna, Austria. Cited on page 103.

Bradford, K., Carpenter, R., and Shaw, B. (2010). Forecasting southern plains wind ramp events using the wrf model at 3-km. In *Proceedings of the AMS Student Conference*, Atlanta, Georgia. Cited on pages 103 and 110.

Buchman, S., Lee, A., and Schafer, C. (2011). High-dimensional density estimation via SCA: An example in the modelling of hurricane tracks. *Statistical Methodology*, 8(1):18–30. Cited on page 126.

Burt, P. and Adelson, E. (1983). The Laplacian Pyramid as a compact image code. *IEEE Transactions on Communications*, 31:532–540. Cited on page 128.

Cawley, G. and Talbot, N. (2004). Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural Networks*, 17(10):1467–1475. Cited on page 132.

Chapelle, O., Vapnik, V., Bousquet, O., and Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159. Cited on page 133.

Chung, F. (1997). Spectral graph theory. In *CBMS Regional Conference Series in Mathematics*, number 92. American Mathematical Society. Cited on page 48.

Coifman, R. and Hirn, M. J. (2013). Diffusion Maps for changing data. *Applied and Computational Harmonic Analysis*, 36(1):79–107. Cited on pages 43 and 56.

Coifman, R. and Lafon, S. (2006a). Diffusion Maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30. Cited on pages 9, 43, 44, 47, 53, 54 and 59.

Coifman, R. and Lafon, S. (2006b). Geometric harmonics: A novel tool for multiscale out-of-sample extension of empirical functions. *Applied and Computational Harmonic Analysis*, 21(1):31–52. Cited on page 126.

Coifman, R., Lafon, S., Lee, A., Maggioni, N., Nadler, B., Warner, F., and Zucker, S. (2005). Geometric Diffusions as a tool for harmonic analysis and structure definition of data: Diffusion Maps. *Proceedings of the National Academy of Sciences of the United States of America*, 102:7426–7431. Cited on pages 43, 47 and 48.

Cox, T. and Cox, M. (2000). *Multidimensional Scaling*. Chapman and Hall/CRC. Cited on pages 6 and 43.

Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning - ICML 2006*, pages 233–240, Pittsburgh, Pennsylvania. ACM. Cited on page 106.

DeLong, E., DeLong, D., and Clarke-Pearson, D. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, 44:837–845. Cited on pages 107 and 113.

Díaz, J., Dorronsoro, J., Fernández, A., Fresnillo, S., Huelin, T., and Valera, A. (2009). A customer management system for the spanish transport system operator. In *11th Spanish-Portuguese Conference on Electrical Engineering - 11 CHLIE 2009*. AEDIE. Cited on page 163.

Do, M. and Vetterli, M. (2003). Framing pyramids. *IEEE Transactions on Signal Processing*, 51:2329–2342. Cited on page 129.

Donoho, D. and Grimes, C. (2003). Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences of the United States of America*, 100(10):5591–5596. Cited on page 43.

Dsilva, C., Talmon, R., Rabin, N., Coifman, R., and Kevrekidis, I. (2013). Nonlinear intrinsic variables and state reconstruction in multiscale simulations. *Journal Chemical Physics*, 139(18). Cited on page 126.

Duda, R., Hart, P., and Stork, D. (2001). *Pattern Classification*. Wiley, New York. Cited on pages 3, 4 and 126.

ECMWF (2005). European Center for Medium–range Weather Forecasts. http://www.ecmwf.int/. Cited on page 56.

Elisseeff, A. and Pontil, M. (2002). Leave-one-out error and stability of learning algorithms with applications. In Suykens, J., Horvath, G., Basu, S., Micchelli, C., and Vandewalle, J., editors, *Advances in learning theory: methods, models and applications*, NATO ASI. IOS Press, Amsterdam; Washington, DC. Cited on page 132.

Emrich, T., Graf, F., Kriegel, H.-P., Schubert, M., Thoma, M., and Cavallaro, A. (2010). CT slice localization via instance-based regression. In *Proceedings of SPIE, Medical Imaging 2010: Image Processing - SPIE 2010*, San Diego, California, USA. Cited on pages 91, 93 and 95.

Fernández, A. (2010). Advanced methods for dimensionality reduction and clustering: Laplacian eigenmaps and spectral clustering. Master's thesis, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Madrid, Spain. Under the supervision of Julia Díaz. Cited on page 161.

Fernández, A., Alaíz, C., González, A., Díaz, J., and Dorronsoro, J. (2012a). Diffusion maps and local models for wind power prediction. In *Artificial Neural Networks and Machine Learning - ICANN 2012*, volume 7553 of *Lecture Notes in Computer Science*, pages 565–572, Heidelberg, Germany. ENNS, Springer-Verlag GmbH. Cited on page 162.

Fernández, A., Alaíz, C., González, A., Díaz, J., and Dorronsoro, J. (2013a). Diffusion maps for wind power ramp detection. In *Advances in Computational Intelligence - IWANN 2013*, volume 7902 of *Lecture Notes in Computer Science*, pages 106–113, Heidelberg, Germany. UMA and UGR and UPC and ULL, Springer-Verlag GmbH. Cited on page 162.

Fernández, A., Alaíz, C., González, A., Díaz, J., and Dorronsoro, J. (2013b). Local anisotropic diffusion detection of wind ramps. Neural Information Processing Systems - NIPS 2013 Workshop: Machine Learning for Sustainability. Cited on page 162.

Fernández, A., González, A., Díaz, J., and Dorronsoro, J. (2012b). Diffusion maps for the description of meteorological data. In *Hybrid Artificial Intelligent Systems - HAIS 2012*, volume 7208 of *Lecture Notes in Computer Science*, pages 276–287, Heidelberg, Germany. Springer-Verlag GmbH. Cited on page 161.

Fernández, A., González, A., Díaz, J., and Dorronsoro, J. (2014a). Diffusion maps for dimensionality reduction and visualization of meteorological data. Submitted to Neurocomputing (Special Issue HAIS 2012). Cited on page 162.

Fernández, A., Rabin, N., Coifman, R., and Eckstein, J. (2014b). Diffusion methods for aligning medical data: Location prediction in CT scan images. *Medical Image Analysis*, 18(2):425–432. Cited on page 162.

Fernández, A., Rabin, N., Fishelov, D., and Dorronsoro, J. (2014). Auto-adaptative laplacian pyramids for high-dimensional data analysis. http://arxiv.org/abs/1311.6594. Cited on page 163.

Ferreira, C., Gama, J., Costa, V. S., Miranda, V., and Botterud, A. (2012). Predicting ramp events with a stream-based HMM framework. In *Discovery Science*, volume 7569 of *Lecture Notes in Computer Science*, pages 224–238. Springer Berlin Heidelberg. Cited on page 103.

Ferreira, C., Gama, J., Matias, L., Botterud, A., and Wang, J. (2011). A survey on wind power ramp forecasting. Technical report, Argonne National Laboratory. Cited on pages 103 and 104.

Fishelov, D. (1990). A new vortex scheme for viscous flows. *Journal of Computational Physics*, 86(1):211–224. Cited on page 131.

Fouss, F., Pirotte, A., Renders, J., and Saerens, M. (2007). Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Enineering*, 19(3):355–369. Cited on page 40.

Frank, A. and Asuncion, A. (2010). UCI machine learning repository. http://archive.ics.uci.edu/ml. Cited on page 98.

Fukunaga, K. and Hummels, D. (1989). Leave-one-out procedures for nonparametric error estimates. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(4):421–423. Cited on page 132.

GAA (2014). Grupo de Aprendizaje Automático de la Universidad Autónoma de Madrid - Software. http://arantxa.ii.uam.es/~gaa/software.html. Cited on page 10.

Gala, Y., Fernández, A., Díaz, J., and Dorronsoro, J. (2013). Support vector forecasting of solar radiation values. In *Hybrid Artificial Intelligent Systems - HAIS 2013*, volume 8073 of *Lecture Notes in Computer Science*, pages 51–60, Heidelberg, Germany. Springer Berlin Heidelberg. Cited on page 163.

Gala, Y., Fernández, A., Díaz, J., and Dorronsoro, J. (2014a). Hybrid machine learning forecasting of solar radiation values. Submitted to Neurocomputing (Special Issue HAIS 2013). Cited on page 164.

Gala, Y., Fernández, A., Dorronsoro, J., García, M., and Rodríguez, C. (2014b). Machine learning prediction of global photovoltaic energy in spain. In *International Conference on Renewable Energies and Power Quality - ICREPQ 2014*, number 12–423. Renewable Energy and Power Quality Journal (RE&PQJ). Cited on page 164.

GEFS (2014). NOAA Global Ensemble Forecast System. http://www.emc.ncep.noaa.gov/GEFS/. Cited on page 144.

GFS (2014). NOAA Global Forecast System. http://www.emc.ncep.noaa.gov/index.php?branch=GFS. Cited on page 70.

Graf, F., Kriegel, H.-P., Pölsterl, S., Schubert, M., and Cavallaro, A. (2011). Position prediction in CT volume scans. In *Proceedings of the 28th International Conference on Machine Learning Workshop on Learning for Global Challenges*, Bellevue, Washington, WA. Cited on pages 91, 93 and 95.

Greaves, B., Collins, J., Parkes, J., and Tindal, A. (2009). Temporal forecast uncertainty for ramp events. *Wind Engineering*, 33(4):309–319. Cited on page 103.

Grimmett, G. and Stirzaker, D. (2001). *Probability and random processes*. Oxford university press. Cited on page 49.

Hagen, L. and Kahng, A. (1992). New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on CAD of Integrated Circuits and Systems*, 11(9):1074–1085. Cited on page 28.

Hamerly, G. and Elkan, C. (2003). Learning the k in k-means. In *Advances in Neural Information Processing Systems 15 - NIPS 2003*, Vancouver, Canada. Cited on pages 57, 155 and 159.

Hastie, T., Tibshirani, R., and Friedman, J. (2008). *The elements of statistical learning: data mining, inference and prediction*. Springer. Cited on pages 88 and 132.

Haupt, S., Mahoney, W., and Parks, K. (2014). Wind power forecasting. In Troccoli, A., Dubus, L., and Haupt, S., editors, *Weather Matters for Energy*, pages 295–318. Springer New York. Cited on page 103.

Hoerl, A. and Kennard, R. (1970). Ridge Regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(12):55–67. Cited on page 73.

Horn, R. and Johnson, C. (1985). *Matrix Analysis*. Cambridge University Press. Cited on page 50.

Hurwitz, J., Nugent, A., Halper, F., and Kaufman, M. (2013). *Big Data For Dummies*. For Dummies. Cited on page 2.

Indyk, P. (2000). Dimensionality reduction techniques for proximity problems. In Shmoys, D., editor, *Symposium on Discrete Algorithms - SODA 2000*, pages 371–378. ACM/SIAM. Cited on page 21.

Jain, A. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666. Cited on page 4.

Jain, A. (2013). Clustering big data. http://www.cse.nd.edu/Fu_Prize_Seminars/jain/slides.pdf. EPS–UAM Seminar celebrated on June 3 2013. Cited on page 3.

Jain, A. and Dubes, R. (1988). *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA. Cited on page 3.

Jolliffe, I. (2002). *Principal Component Analysis*. Springer Series in Statistics. Cited on pages 6 and 43.

Kaggle (2014). American meteorological society 2013–2014 solar energy prediction contest. https://www.kaggle.com/c/ams-2014-solar-energy-prediction-contest. Cited on page 144.

Kamath, C. (2010). Understanding wind ramp events through analysis of historical data. In *Transmission and Distribution Conference and Exposition*, 2010 IEEE PES, pages 1–6. Cited on page 103.

Kamath, C. (2011). Associating weather conditions with ramp events in wind power generation. In *Proceedings of the Power Systems Conference and Exposition*, 2010 IEEE PES, Phoenix Convention Center, AZ, USA. Cited on page 103.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - IJCAI 1995*, pages 1137–1143, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. Cited on page 132.

Kohonen, T. (1988). Neurocomputing: Foundations of research. chapter Self-organized Formation of Topologically Correct Feature Maps, pages 509–521. MIT Press, Cambridge, MA, USA. Cited on page 6.

Kruskal, J. and Wish, M. (1978). *Multidimensional scaling*, volume 11. Sage. Cited on page 43.

Kushnir, D., Haddad, A., and Coifman, R. (2012). Anisotropic Diffusion on sub-manifolds with application to earth structure classification. *Applied and Computational Harmonic Analysis*, 32(2):280–294. Cited on page 90.

Lafon, S. (2004). *Diffusion maps and geometric harmonics*. PhD thesis, Yale University. Cited on page 47.

Laney, D. (2001). 3d data management: Controlling data volume, velocity and variety. Technical report, Meta Group. Cited on page 2.

Liu, L., Gan, L., and Tran, T. (2008). Lifting-based laplacian pyramid reconstruction schemes. In *ICIP*, pages 2812–2815. IEEE. Cited on page 129.

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137. Cited on page 4.

Lorenz, E. (1963). Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2):130–141. Cited on page 107.

Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416. Cited on pages 9, 25, 33 and 44.

Luxburg, U., Belkin, M., and O.Bousquet (2008). Consistency of spectral clustering. *Annals of Statistics*, 36(2):555–586. Cited on page 42.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14. California, USA. Cited on page 4.

Meilă, M. (2006). The uniqueness of a good optimum for k-means. In *Proceedings of the 23rd International Conference on Machine Learning - ICML 2006*, pages 625–632. ACM. Cited on page 4.

Mishne, G. and Cohen, I. (2013). Multiscale anomaly detection using diffusion maps. *Journal of Selected Topics in Signal Processing*, 7(1):111–123. Cited on page 129.

Mohanty, S., Jagadeesh, M., and Srivatsa, H. (2013). *Big Data Imperatives: Enterprise Big Data Warehouse, BI Implementations and Analytics*. Apress, Berkely, CA, USA. Cited on page 2.

Mouysset, S., Guivarch, R., Noailles, J., and Ruiz, D. (2012). Parallel spectral clustering for the segmentation of cdna microarray images. In *International Conference on Practical Applications of Computational Biology & Bioinformatics*, volume 154, pages 1–9. Springer. Cited on page 54.

Nadler, B., Lafon, S., Coifman, R., and Kevrekidis, I. (2005). Diffusion Maps, Spectral Clustering and eigenfunctions of Fokker–Planck operators. In *Advances In Neural Information Processing Systems 18 - NIPS 2005*, Vancouver, Canada. Cited on page 43.

Newman, M. (2010). *Networks: An Introduction*. Oxford University Press, Inc., New York. Cited on pages 45 and 51.

Ng, A., Jordan, M., and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 13 - NIPS 2001*, pages 849–856. MIT Press. Cited on page 44.

Peña, J. M., Lozano, J. A., and Larrañaga, P. (1999). An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, 20(10):1027–1040. Cited on page 60.

Pinson, P., Nielsen, H., Madsen, H., and Nielsen, T. (2009). Local linear regression with adaptive orthogonal fitting for the wind power application. *Statistics and Computing*, 18(1):59–71. Cited on page 70.

Rabin, N. (2010). *Data Mining in Dynamically Evolving Systems via Diffusion Methodologies*. PhD thesis, Tel-Aviv University. Cited on pages 54 and 58.

Rabin, N. and Coifman, R. (2012). Heterogeneous datasets representation and learning using Diffusion Maps and Laplacian Pyramids. In *Proceedings of the 12th SIAM International Conference on Data Mining - SDM 2012*, Anaheim, California, USA. Cited on pages 55, 125, 126, 129 and 133.

Rasmussen, C. and Williams, C. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press. Cited on page 133.

Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J., and Müller, M. (2011). pROC: an open-source package for R and S+ to analyze and compare ROC curves. *BMC Bioinformatics*, 12:77. Cited on page 113.

Rogers, L. and Williams, D. (2000a). *Diffusions, Markov Processes, and Martingales. Vol. 1: Foundations*. Cambridge University Press, Cambridge. Cited on pages 44, 49 and 84.

Rogers, L. and Williams, D. (2000b). *Diffusions, Markov Processes, and Martingales. Vol. 2: Itô Calculus*. Cambridge University Press, Cambridge. Cited on page 85.

Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326. Cited on pages 9, 22, 23 and 43.

Saul, L. and Roweis, S. (2003). Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155. Cited on page 43.

Seung, H. and Lee, D. (2000). The manifold ways of perception. *Science*, 290(5500):2268–2269. Cited on page 43.

Sevlian, R. and Rajagopal, R. (2012). Wind power ramps: Detection and statistics. In *Power and Energy Society General Meeting*, 2012 IEEE PES, pages 1–8. Cited on page 103.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):888–905. Cited on pages 28 and 44.

Simon, P. (2013). *Too Big to Ignore: The Business Case for Big Data*. John Wiley & Sons. Cited on page 2.

Singer, A. and Coifman, R. (2008). Non-linear Independent Component Analysis with Diffusion Maps. *Applied and Computational Harmonic Analysis*, 25(2):226–239. Cited on pages 82, 83, 85, 86 and 88.

Sotavento (2013). Sotavento Galicia wind farm. http://www.sotaventogalicia.com/index.php. Cited on page 109.

Soumen, C., Martin, E., Usama, F., Johannes, G., Jiawei, H., Gregory, M. S. P., and Wei, W. (2006). Data mining curriculum: A proposal (version 1.0). http://www.kdd.org/sites/default/files/CURMay06.pdf. Cited on page 3.

Steinhaus, H. (1956). Sur la division des corps matériels en parties. *Bull. Acad. Polon. Sci. Cl. III. 4*, pages 801–804. Cited on page 4.

Szlam, A., Maggioni, M., and Coifman, R. (2008). Regularization on graphs with function-adapted diffusion processes. *Journal of Machine Learning Research*, 9:1711–1739. Cited on pages 155 and 159.

Talmon, R., Cohen, I., and Gannot, S. (2011). Supervised source localization using diffusion kernels. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics - WASPAA 2011*, pages 245–248. IEEE. Cited on page 90.

Talmon, R., Cohen, I., Gannot, S., and Coifman, R. (2013). Diffusion maps for signal processing: A deeper look at manifold-learning techniques based on kernels and graphs. *IEEE Signal Process. Mag.*, 30(4):75–86. Cited on page 90.

Tenenbaum, J., Silva, V., and Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323. Cited on pages 6 and 43.

Wagner, D. and Wagner, F. (1993). Between min cut and graph bisection. In *Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science*, pages 744–750, London, UK. Springer-Verlag. Cited on page 29.

Williams, C. and Seeger, M. (2001). Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13 - NIPS 2001*, pages 682–688. MIT Press. Cited on page 127.

Xu, R., Damelin, S., Nadler, B., and D.C. Wunsch, I. (2010). Clustering of high-dimensional gene expression data with feature filtering methods and Diffusion Maps. *Artificial Intelligence in Medicine*, 48(2-3):91–98. Cited on page 56.

Yu, W., Dyck, S., Plante, A., Choisnard, J., and Forcione, A. (2013). Forecast of wind ramps in gaspe region of quebec. In *EGU General Assembly Conference Abstracts*, volume 15 of *EGU General Assembly Conference Abstracts*, page 12797. Cited on page 103.

Zack, J., Young, S., Cote, M., and Nocera, J. (2010). Development and testing of an innovative short-term large wind ramp forecasting system. In *Proceedings of the European Wind Energy Conference and Exhibition - EWEC 2010*, Warsaw, Poland. EWEA. Cited on page 103.

Zheng, H. and Kusiak, A. (2009). Prediction of wind farm power ramp rates: A data-mining approach. *Journal of Solar Energy Engineering*, 131(3):031011–1–031011–8. Cited on page 103.